PROGRAMACION EN C: BLOQUE1

1) PROGRAMA HOLA MUNDO:

COMENTARIOS.

Comenzamos siempre con un comentario como vemos en la linea 1 (1 //Programa Hola mundo). Los comentarios nos facilitan saber de que trata el programa .

LIBRERIAS.

Siempre solemos cargar una librería en este caso : #include<stdio.h>

MAIN.

En este caso (int main( ){……lo que haga}). El main es donde se ejecutan las acciones del programa. Finaliza con un return 0.

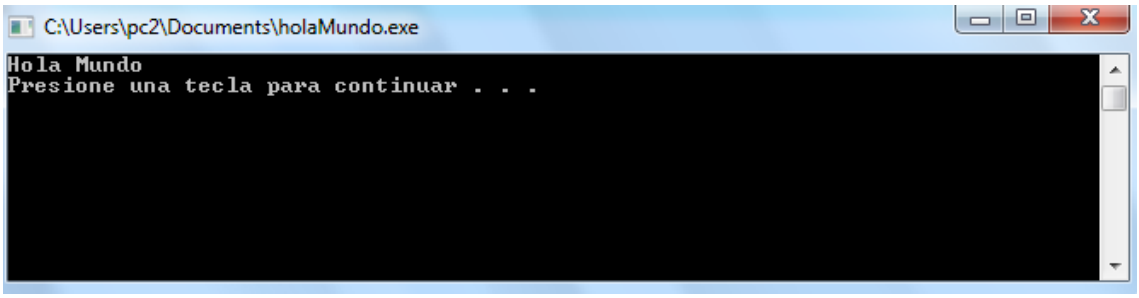
Funcion printf ( ); :

Esta funcion nos permite mostrar por pantalla algo , en este caso la frase : Hola mundo
que esta entre comillas por se unas frase (String).

Salto de linea :

Ponemos \n. no tiene más complicacion.

```
1   //Programa Hola mundo
2
3   #include<stdio.h>
4
5   int main()
6   {
7       printf ("Hola Mundo\n");
8
9       system  ("pause");// introduce una tecla .es recomendable
10      return 0;
11  }
12
```



2) VARIABLES:

Entendemos como variable : la forma de ponerle un nombre a un espacio de la memoria donde guarda un dato.

Las variables siempre deben de ser declaradas SIEMPRE y hay varios tipos y podemos inicializarlas para después imprimirlas.

```c
//Programa Variables

#include<stdio.h>

int main()
{
        //DECLARAMOS VARIABLES:

    int x;     // variable declarada de tipo entero (int) como 8,0,-9....
    float y;   // variable declarada de tipo flotante-decimal (float) como 9.67 , 5.1 ...
    double y2; // igual que la anterior pero el double es para numeros decimales más grandes
    char z;    // variable de tipo caracter (char) como a ,b,c,d.....,z

        //INICIALIZAMOS VARIABLES:

    x = 5;     // variable inicializada a 5 ; x toma como valor entero 5.
    y = 2.4;   // variable inicializada a 2.4 ; x toma como valor decimal 2.4.
    y2 = 3.4;  // igual que el anterior pero y2 ahora vale 3,4
    z = 'a';   // variable inicializada al caracter a; z vale la letra a .

        //IMPRIMIMOS VALORES :

    printf ("Valor de x: %i.\n",x); // para imprimir una variable ponemos
    //  %i. ( la i es de entero(int)) y despues ponemos ,x

    printf ("Valor de y: %f.\n",y);     //  %f. ( la f es de flotante(float)) y despues ponemos ,y
    printf ("Valor de y2: %f.\n",y2);   //  %f. ( se pone %f. no (%d.) porque sigue siendo un decimal
    printf ("Valor de z: %c.\n",z);     //  %c. ( la c es de caracter(char)) y despues ponemos ,z)

    system  ("pause");
    return 0;
}

```

También podemos imprimir las partes del printf( ) todo junto respetando el orden claro:

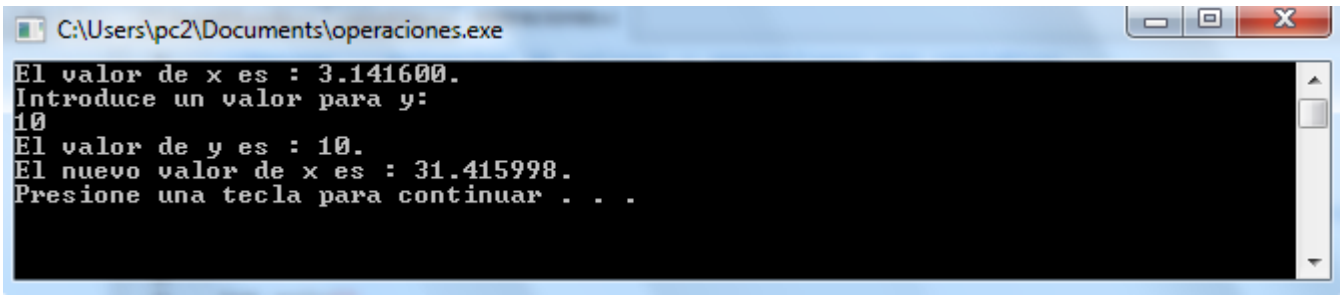
```c
    //IMPRIMIMOS VALORES :

printf ("Valor de x: %i.\nValor de y: %f.\nValor de y2: %f.\nValor de z: %c.\n",x,y,y2,z);
```

El resultado es el mismo.

3) MARCOS , ASIGNACION DE VALORES , SCANNER Y OPERACIONES CON VARIABLES

```c
//Marcos  , asinacion de valores y operaciones con variables

#include<stdio.h>
#define PI 3.1416 // ESTO ES UNA MACRO:
//Una macro es solamente que cada vez que apareca la palabra (en este caso PI)
// entenderá que vale (3.1416) pero pude ser otras palabras con diferentes valores
// el que se quiera poner da igual.

int main()
{
    //DECLARAMOS
    float x;
    int y;
    //AISGNAMOS
    x = PI;

    //IMPRIMIMOS
    printf ("El valor de x es : %f.\n",x);

    // hagamos que un programa nos pida una variable :

    printf ("Introduce un valor para y: \n");
    scanf("%i",&y); // metemos por scanner un valor para y.
    printf ("El valor de y es : %i.\n",y); // mustra el valor metido para y.

    // operaciones:
    x = x * y;
    printf ("El nuevo valor de x es : %f.\n",x);

    system  ("pause");
    return 0;
}
```

COMO RESULTADO EL PROGRAMA AL SER EJECUTADO NOS DARÁ:



4) AMPLIANDO CONOCIMIENTOS SOBRE ASIGNACION Y OPERACIONES BÁSICAS.

```c
// AMPLIACION CONOCIMIENTOS , ASIGNACION Y OPERACIONES BÁSICAS:

#include <stdio.h>
#include <math.h>
int main () {
    int x,y,z; // declaramos 2 enteros

    x=10;  //inicializamos x
    y=2;   //inicializamos y

    //OPERACIONES BÁSICAS:
    z = x + y;                                  //--------> SUMA
    printf ("El valor de x + y es:  %i.\n\n",z);
    z = x - y;                                  //--------> RESTA
    printf ("El valor de x - y es:  %i.\n\n",z);
    z = x * y;                                  //--------> PRODUCTO
    printf ("El valor de x * y es:  %i.\n\n",z);
    z = x / y;                                  //--------> DIVISION
    printf ("El valor de x / y es:  %i.\n\n",z);
    z = x % y;                                  //--------> RESTO DE LA DIVISION
    printf ("El resto de la division x entre y es:  %i.\n\n",z);
    //OPERACIONES BASICAS CON MATH:
    x = pow (y,x);                                //--------> POTENCIA
    printf ("El valor de y elevado a x es:  %i.\n\n",x);
    x = sqrt (x);                               //--------> RAIZ
    printf ("la raiz cuadrada de x es:  %i.\n\n",x);
    system ("pause");
    return 0;
}

```

```
El valor de x + y es:  12.

El valor de x - y es:  8.

El valor de x * y es:  20.

El valor de x / y es:  5.

El resto de la division x entre y es:  0.

El valor de y elevado a x es:  1024.

la raiz cuadrada de x es:  32.

Presione una tecla para continuar . . .
```

Debemos saber que podemos simplificar la forma de declarar las operaciones:

Si tuviéramos : x = x + y; ---------- x += y; x = x - y; ---------- x -= y;

x = x * y; ---------- x *= y; x = x / y; ---------- x /= y;

operadores de incremento y decremento:

x++; significa que el valor de x aumenta una unidad || x--; significa que el valor de x disminuye una unidad.

5) METODO DE LA BURBUJA:

INTERCAMBIA EL VALOR DE 2 NUMEROS ENTEROS QUE INTRODUCIMOS POR TECLADO.

```c
// metodo de la burbuja

#include <stdio.h>

int main (){

    int x, y, aux; // aux es una variable auxiliar, que se usara en este metodo.

    printf ("El valor de x es : "); // INTRODUCIMOS EL VALOR DE X POR TECLADO
    scanf ("%i",&x);

    printf ("El valor de y es : "); // INTRODUCIMOS EL VALOR DE Y POR TECLADO
    scanf ("%i",&y);

    aux = x; // En aux metemos el valor de una de las variables
    x= y; // a la variable que le hemos asignado a auxiliar le metemos el valor de la
    // otra variable .El valor original de la variable x que lo hemos perdido
    //ya y que necesita "y" lo tenemos guardado en auxiliar POR TANTO :
    y = aux;

    printf ("El valor de x es: %i.\n",x);
    printf ("El valor de y es: %i.\n",y);

    system ("pause");
    return 0;
    }
```

```
C:\Users\pc2\Documents\metodo burbuja.exe
El valor de x es : 5
El valor de y es : 10
El valor de x es: 10.
El valor de y es: 5.
Presione una tecla para continuar . . .
```

PROGRAMACION EN C: BLOQUE 2

1) ESTRUCTURAS SELECTIVAS :

Basicamente trata sobre :

"si se cumple una condicion haz una cosa ; si no se cumple haz otra"

if(condicion)

{ejecuta acción }

else

{ejecuta otra accion}

OPERADORES : x == y ---x es igual a y ; x < y ---x menor que y ; x <= y ----x menor o igual que y

x > y ---x es mayor que y ; x >= y ---x mayor o igual que y ; x != y ----x distinto que y

```c
//ESTRUCTURAS SELECTIVAS 1

#include <stdio.h>

int main (){
    int x;

    x=5;
       //COMIENZO DE LA ESTRUCTURA SELECTIVA.
    if(x == 5)  {                           // si x es igual a 5 (CONDICION).
        printf ("El valor de x es 5.\n");     // entonces se ejecuta esta linea siempre y cuando x = 5.
    }
    else{                                   // si no
        printf ("El valor de x  NO es 5.\n"); //ejecuta la condicion de esta linea.
    }

    system ("pause");
    return 0;
    }

```

PROBLEMA : declarar una variable de tipo int y la iniciamos con un valor entre 1 y 10 por ejemplo 9 por teclado solicitamos un numero y vemos si hemos acertado y dejamos tres intentos.

```c
#include <stdio.h>
int main (){
    int x , y;

    x=9;
    printf ("introduce un valor para y : "); // INTRODUCE UN VALOR PARA Y
    scanf ("%i",&y);                         // INTRODUCIMOS EL VALOR DE Y POR TECLADO
    //COMIENZO DE LA ESTRUCTURA SELECTIVA.-----------------------------------------
    if(x == y)  {printf ("ACERTASTE\n\n");}  //si x = y , imprime ACERTASTE
    else{                                    // si no ...
        printf ("TU NUMERO NO ES EL MIO\n\n intentalo de nuevo"); //x es distinto de y e imprime TU NU...
        scanf ("%i",&y);                     // vuelve a introducir otro valor para y
        if(x == y)  {printf ("ACERTASTE\n\n"); }   // se repite la estructura selectiva...
        else{
            printf ("TU NUMERO NO ES EL MIO\n\n intentalo de nuevo");
            scanf ("%i",&y);}
            if(x == y)  {printf ("ACERTASTE\n\n"); }
            else{printf ("TU NUMERO NO ES EL MIO\n\n fallaste los tres intentos :(\n\n)");}
        //pasados tres intentos el programa termina.
    }
    system ("pause");
    return 0;
    }
```

```
C:\Users\pc2\Documents\selectivas2.exe
introduce un valor para y : 3
TU NUMERO NO ES EL MIO

 intentalo de nuevo5
TU NUMERO NO ES EL MIO

 intentalo de nuevo6
TU NUMERO NO ES EL MIO

 fallaste los tres intentos :(

)Presione una tecla para continuar . . .
```

En este caso en los tre intentos hemos introducido 3 ,5,6 los cuales no son 9 por tanto se fallan los 3 intentos.

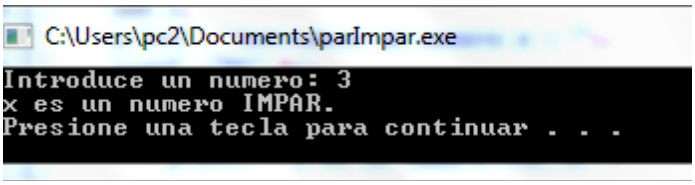
```
C:\Users\pc2\Documents\selectivas2.exe
introduce un valor para y : 9
ACERTASTE

Presione una tecla para continuar . . .
```

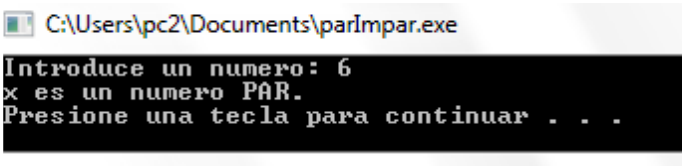
Ahora hemos introducido el 9 y como es el numero que vale x , HEMOS ACERTADO.

2) VER SI UN NUMERO ES PAR Y HALLAR EL MAXIMO DE 3 NUMEROS:

Queremos realizar un programa que lea de la entrada estandar un numero entero positivo y escriba en la salida estandar si es par o impar.

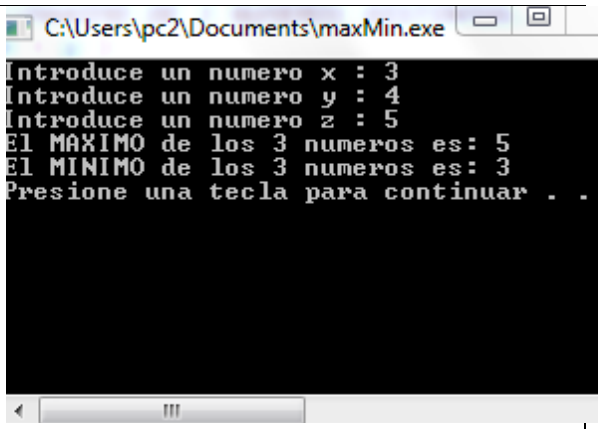
```c
#include <stdio.h>

int main (){
    int x;
    printf ("Introduce un numero x : ");
    scanf ("%i",&x);
       //COMIENZO DE LA ESTRUCTURA SELECTIVA.
    if(x%2==0)  {                           // si x es igual a 5 (CONDICION).
        printf ("x es un numero PAR.\n");     // entonces se ejecuta esta linea siempre y cuando x = 5.
    }
    else{                                   // si no
        printf ("x es un numero IMPAR.\n"); //ejecuta la condicion de esta linea.
    }

    system ("pause");
    return 0;
    }
```





Realizar un programa que lea 3 numeros distintos e imprima el maximo y el minimo que vale de los 3

```c
#include <stdio.h>
int main (){
    int x ,y ,z;
    int max, min;
    printf ("Introduce un numero x : ");
    scanf ("%i",&x);
    printf ("Introduce un numero y : ");
    scanf ("%i",&y);
    printf ("Introduce un numero z : ");
    scanf ("%i",&z);

       //MAXIMO DE 3 NUMEROS DISTINTOS.
    if(x > y )
        {
          if(x > z ){max = x;}
          else{max = z;}
        }
    else{
        if(y > z ){max = y;}
        else{max = z;}
        }
        //MINIMO DE 3 NUMEROS DISTINTOS.
    if(x < y )
        {
          if(x < z ){min = x;}
          else{min = z;}
        }
    else{
        if(y < z ){min = y;}
        else{min = z;}
        }
    printf ("El MAXIMO de los 3 numeros es: %i\n",max);
    printf ("El MINIMO de los 3 numeros es: %i\n",min);

    system ("pause");
    return 0;
    }
```



3) ESTRUCTURAS IF /ELSE ANIDADAS Y SWITCH:

Realiza un programa que lea un número entre 1 y 7 y devuelva el día de la semana correspondiente

(1 para lunes ,2para martes ...).

LOS 2 CUADROS DE ARRIBA Y DEBAJO DE LA IZQUERDA HACEN LO MISMO PERO ESCRITOS DE DIFERENTES MANERAS

```c
#include <stdio.h>
int main (){

    int x;
    printf ("introduce un numero entre 1 y 7: ");
    scanf ("%i",&x);
    if(x == 1){printf ("LUNES.\n");}
    else if(x == 2){printf ("MARTES.\n");}
    else if(x == 3){printf ("MIERCOLES.\n");}
    else if(x == 4){printf ("JUEVES.\n");}
    else if(x == 5){printf ("VIERNES.\n");}
    else if(x == 6){printf ("SABADO.\n");}
    else if(x == 7){printf ("DOMINGO.\n");}
    else{printf ("NUMERO INCORRECTO.\n");}
    system ("pause");
    return 0;
    }
```

```
C:\Users\pc2\Documents\semana.exe
introduce un numero entre 1 y 7: 5
VIERNES.
Presione una tecla para continuar . . .
```

if/else if/else:
si/si no/resto casos:

if(condición1){acción1;}
else if(condición2){acción2;}
……
else if(condición N){acción N;}
else{acción resto casos;}

```c
#include <stdio.h>
int main (){

    int x;
    printf ("introduce un numero entre 1 y 7: ");
    scanf ("%i",&x);

    switch (x){
        case 1: printf ("LUNES.\n");break;
        case 2: printf ("MARTES.\n");break;
        case 3: printf ("MIERCOLES.\n");break;
        case 4: printf ("JUEVES.\n");break;
        case 5: printf ("VIERNES.\n");break;
        case 6: printf ("SABADO.\n");break;
        case 7: printf ("DOMINGO.\n");break;
        default: printf ("NUMERO INCORRECTO.\n");
    }

    system ("pause");
    return 0;
    }
```

```
C:\Users\pc2\Documents\semanasSwitch.exe
introduce un numero entre 1 y 7: 4
JUEVES.
Presione una tecla para continuar . . .
```

switch: (mas eficientes que if/else if/else)
switch (x){
case 1: acción 1 ;break;
case 2: acción 2 ;break;
……..
case n: acción N ;break;
default: acción en cualquier otro caso;
}

Español:
Para x
x = caso 1 ejecuta acción 1
sino
x = caso 2 ejecuta acción 2
sino
…….
x = caso N ejecuta acción N
sino
caso defecto en que x no es ningún caso anterior
ejecuta accion en cualquier otro caso

4) CONDICIONES MULTIPLES:

Las condiciones multiples son básicamente if/else o if/if else/else donde la condición principal está compuesta de dos o mas condiciones:

Ejemplo &&: if(x>=1 && x<=10) ----------si x es mayor o igual que 1 Y x es menor o igual que 10

(las 2 condiciones deben ser cumplidas para que se ejecute la acción que realicen.)

Ejemplo ||: if(x>=1 || x<=10) ----------si x es mayor o igual que 1 o x es menor o igual que 10

(al menos una de las condiciones deben ser cumplida para que se ejecute la acción que realicen.)

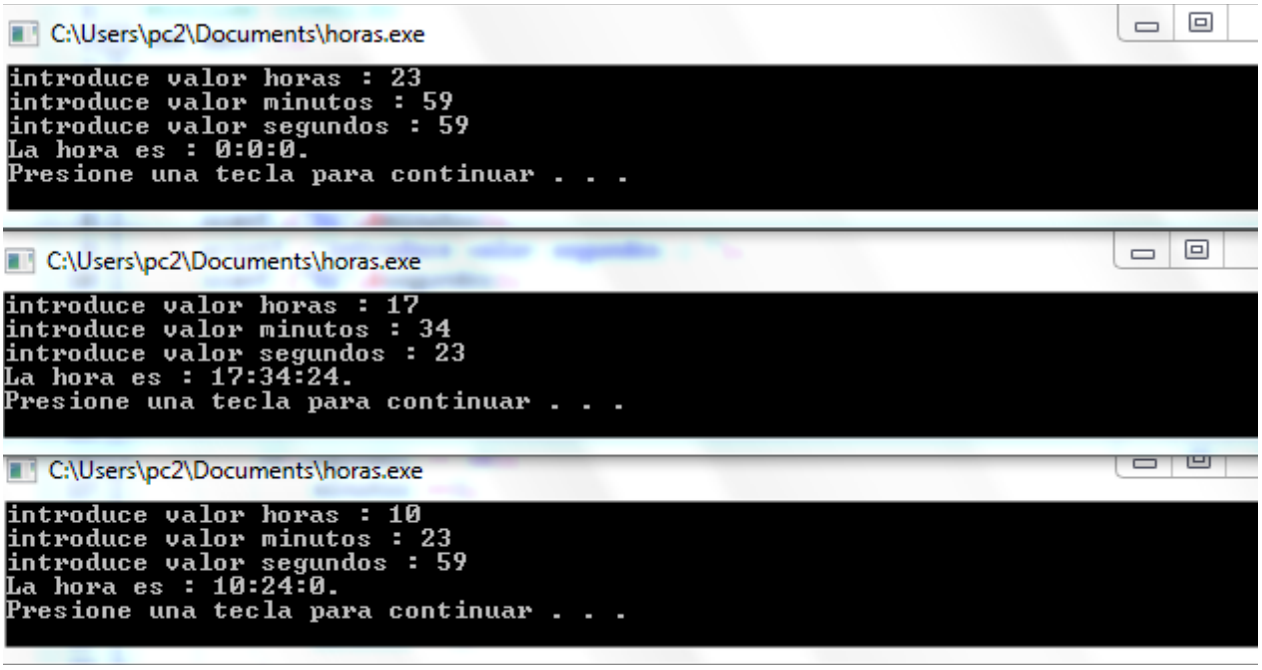
```c
#include <stdio.h>
int main (){

    int x;
    printf ("introduce un numero : ");
    scanf ("%i",&x);

    if (x>=1 && x<=10){
        printf ("El numero %i esta entre 1 y 10.\n:",x);
    }
    else{
        printf ("El numero %i NO esta entre 1 y 10.\n:",x);
    }
    system ("pause");
    return 0;
    }
```

```
C:\Users\pc2\Documents\condicionesMultiples.exe
introduce un numero : 6
El numero 6 esta entre 1 y 10.
:Presione una tecla para continuar . . .
```

```
C:\Users\pc2\Documents\condicionesMultiples.exe
introduce un numero : 12
El numero 12 NO esta entre 1 y 10.
:Presione una tecla para continuar . . .
```

EJERCICIO: escribir un programa que acepte 3 números: horas , minutos y segundos y devuelva la hora que será dentro de un segundo ,controlando que sea una hora correcta:

```c
#include <stdio.h>
int main (){

    int horas , minutos , segundos;
    printf ("introduce valor horas : ");
    scanf ("%i",&horas);
    printf ("introduce valor minutos : ");
    scanf ("%i",&minutos);
    printf ("introduce valor segundos : ");
    scanf ("%i",&segundos);


    if (horas<=23 && minutos<=59 && segundos<=59){
        segundos +=1;

        if(segundos == 60){
            minutos +=1;
            segundos = 0;
        }
        if(minutos == 60){
            horas += 1;
            minutos = 0;
        }
        if(horas == 24){
            horas = 0;
        }

        printf ("La hora es : %i:%i:%i.\n",horas,minutos,segundos);
    }
    else{
        printf ("LA HORA INTRODUCIDA NO ES CORRECTA.\n:");
    }
    system ("pause");
    return 0;
    }

```



PROGRAMACION EN C: BLOQUE 3

1) ESTRUCTURAS ITERATIVAS (BUCLES FOR , WHILE y DO-WHILE) :

Los bucles nos permiten realizar una sentencia de codigo mas de una vez mientras se cumpla una condición y recorriendo con un contador , una vez que la condicion no se cumpla se sale del bucle .

BUCLE FOR:	**BUCLE WHILE:**
for (inicializar contador ; condicion ; avance del contador) **{** **Acciones a realizar mientras se cumpla la condicion .** **}**	**inicializar contador** **while (condicion)** **{** **Acciones a realizar mientras se cumpla la condicion .** **avance del contador.** **}**

Una vez vista la estructura de estos bucles programemos un funcionamiento simple de ellos.

EJERCICIO: queremos un programa que lea 2 numeros y muestre por pantalla todos los numeros comprendidos dentro de ese intervalo:

```c
#include<stdio.h>

int main (){

    int x, y, i; // declaramos x , y , i (contador)

    printf ("Introduce un numero: ");
    scanf ("%i",&x);
    printf ("Introduce un numero mayor que el anterior: ");
    scanf ("%i",&y);

    //BUCLE FOR:
    for (i = x + 1 ; i  < y ; i++){  // para i sea el siguiente numero de x ,mientras el i
        printf ("%i, ",i);    // sea estrictamente menor que y, avanza el contador i una unidad .
    }                         //  imprime todos los numeros i que estan dentro de la condicion

    printf ("\n");

    //BUCLE WHILE:
    i = x + 1;                // i sea el siguiente numero de x
    while (i < y){            // mientras el i sea estrictamente menor que y
        printf ("%i, ",i);    //  imprime todos los numeros i que estan dentro de la condicion
        i++;                  // avanza el contador i una unidad .
    }

    system ("pause");
    return 0;
}
```

```
C:\Users\pc2\Documents\forYwhile1.exe
Introduce un numero: 5
Introduce un numero mayor que el anterior: 10
6, 7, 8, 9,
6, 7, 8, 9, Presione una tecla para continuar . . .
```

EJERCICIO FOR Y WHILE:

FOR:
Realizar un programa que imprima la suma de los numeros pares entre 2 y 20

```c
#include<stdio.h>

int main (){

    int i ;
    int suma = 0;

    //BUCLE FOR:
    for (i = 2 ; i  <= 20 ; i+= 2){
        suma += i;
    }
    printf ("%i, ",suma);

    system ("pause");
    return 0;
}
```

```
C:\Users\pc2\Documents\forEjer.exe
110, Presione una tecla para continuar
```

WHILE:
Realizar un programa que calcule la media de x numeros introducidos por teclado

```c
#include<stdio.h>

int main (){

    int x,y,i;
    float suma;
    printf ("Introduce de cuantos numeros quieres calcular la media : ");
    scanf ("%i",&x);

    i = 0;
    suma = 0;
    while(i < x){
        printf ("introduce el %i numero : ",i+1);
        scanf ("%i",&y);
        suma = suma + y;
        i++;
    }
    suma = suma / x;
    printf ("la media de los numeros introducidos es : %f",suma);
    system ("pause");
    return 0;
}
```

```
C:\Users\pc2\Documents\whileEjer.exe
Introduce de cuantos numeros quieres calcular la media : 3
introduce el 1 numero : 5
introduce el 2 numero : 10
introduce el 3 numero : 15
la media de los numeros introducidos es : 10.000000Presione
```

BUCLE DO-WHILE: Siempre se ejecuta almenos una vez antes de la condión:

DO-WHILE

do
{
(accion)
}
while(condicion);

OBSERVACION 1: ffluch (stdin) se pone cada vez que queramos recoger una variable de tipo char metida por teclado y la hagamos mas de una vez se pone.

OBSERVACION 2: ponemos && y no || porque solo el && obliga a que las 2 condiciones sean ciertas (sigue el bucle) de lo contrario saldriamos del bucle

Hacer un bucle que pregunte si quiero continuar y hasta que no escribamos s o S siga preguntando

```c
#include<stdio.h>

int main (){
    char a;

    do{
        printf ("Quieres continuar?: \n");
        printf ("pon s o S o te vuelvo a preguntar: ");
        fflush (stdin); //------------OBSERVACION 1
        scanf ("%c",&a);
        if (a != 's' && a != 'S' ){
        printf ("Esta no es letra s ni S: \n");
    }
    } while(a != 's' && a != 'S'); //--OBSERVACION 2
    system ("pause");
    return 0;
}
```

```
C:\Users\pc2\Documents\doWhileEjer.exe
Quieres continuar?:
pon s o S o te vuelvo a preguntar: r
Esta no es letra s ni S:
Quieres continuar?:
pon s o S o te vuelvo a preguntar: s
Presione una tecla para continuar . . .
```

Otro ejemplo del DO-WHILE: programa que lea 2 numeros 1º un numero y el 2º otro mayor que el primero y muestre por pantalla los numeros comprendidos entre estos.

```c
#include<stdio.h>

int main (){

    int x, y, i;
    printf ("Introduce un num: ");
    scanf ("%i",&x);
    do{
        printf ("Introduce un num > que el anterior: ");
        scanf ("%i",&y);
    }while (y < x);

    for (i = x + 1 ; i  < y ; i++){
        printf ("%i, ",i);
    }
    system ("pause");
    return 0;
}
```

```
C:\Users\pc2\Documents\doWhile2.exe
Introduce un num: 5
Introduce un num > que el anterior: 3
Introduce un num > que el anterior: 8
6, 7, Presione una tecla para continuar . . .
```

Este codigo es muy similar a otro ejemplo que tenemos más arriba pero hay un DETALLE muy importante que es : ¿Qué pasa si el usuario mete por descuido en el segundo numero un valor menor que el primero?

Nosotros como programadores debemos de pensar hasta en los casos mas idiotas, de ahí resultará que realizaremos un codigo eficiente libre de BUGS.

Con esto digo que si miramos el codigo vemos que hay un while que manda entrar al comienzo del do si y solo si el numero y (2º numero) es menor que el x(1º numero), tendremos que volver a introducir otro numero para y hasta que introduzcamos un y mayor que el x .
Si es mayor entramos en el bucle for que realiza el resto .

2) BUCLES ANIDADOS Y EJERCICIO CRONOMETRO :

Los bucles anidados son basicamente un bucle con mas de un bucle dentro .

Ejercicio : programa las tablas de multiplicar del 1 al 9:

```c
#include<stdio.h>

int main (){

    int  i ,j;

    for (i =1 ; i<=9;i++){
        for (j =1 ; j<=10;j++){
            printf ("%i x %i = %i.\n",i,j,i*j);
            // los printf permiten hacer operaciones dentro de ellos

        }
        printf ("\n");
    }

    system ("pause");
    return 0;
}
```

```
1 x 1 = 1.
1 x 2 = 2.
1 x 3 = 3.
1 x 4 = 4.
1 x 5 = 5.
1 x 6 = 6.
1 x 7 = 7.
1 x 8 = 8.
1 x 9 = 9.
1 x 10 = 10.

2 x 1 = 2.
2 x 2 = 4.
2 x 3 = 6.
2 x 4 = 8.
2 x 5 = 10.
2 x 6 = 12.
2 x 7 = 14.
2 x 8 = 16.
2 x 9 = 18.
2 x 10 = 20.

3 x 1 = 3.
3 x 2 = 6.
3 x 3 = 9.
3 x 4 = 12.
3 x 5 = 15.
3 x 6 = 18.
3 x 7 = 21.
```

Ejercicio Cronometro: hacer un cronometro que comienze cuando puelsemos la tecla espacio:

```c
#include<stdio.h>
#include<windows.h>  // para windows
//#include<unistd.h> // para linux
int main (){

     int  h , m, s ,x;

     x= 1000;
     for (h =0 ; h<24;h++){
        for (m =0 ; m<60;m++){
            for (s =0 ; s<60;s++){
            printf ("%02i:%02i:%02i.\r",h,m,s); //  \r sobrescribe la linea por la siguiente iteracion
            // al poner el 02 entre el % y el i , lo unico que hace es que cada variable se
            //componga de 2 numeros, porque en un reloj vemos 00:00:10 no 0:0:10

            Sleep(x);//nuestro programa se duerme durante un periodo de tiempo
            // en este caso x = 1000 que equivalen a 1 seg al ser 1000 milisegundos

             //usleep (1000 * x); PARA LINUX Y SON MICRO SEGUNDOS pero x por 1000 es lo mismo
           }
         }
         printf ("\n");
     }

     system ("pause");
     return 0;
}
```

```
C:\Users\pc2\Documents\buclesAnidados3.exe
00:01:18.
```

3) CALCULO DEL FACTORIAL DE UN NUMERO:

Un numero factorial es basicamente mutiplicar todos los numeros desde el numero que se quiere el factorial hasta el 1 ; **Ejemplo : Calcular el factorial del numero 4: (4!—es como se representa el factorial de 4)**

Se calcula de la siguiente manera : 4! = 4 * 3 * 2 *1 que resulta 4! = 24.

```c
#include <stdio.h>

int main (){
    int x , i, fact;
    printf ("calcula factorial del numero  : ");
    scanf ("%i",&x);
    fact = 1;
    for (i = 1; i<=x ; i++){
        fact = fact * i;
    }
    printf ("el factorial de numero %i es: %i.\n",x,fact);
    system ("pause");
    return 0;
}
```

```
C:\Users\pc2\Documents\factorial.exe
calcula factorial del numero  : 4
el factorial de numero 4 es: 24.
Presione una tecla para continuar . . .
```

4) SERIE DE FIBONACCI :

La serie de fibonacci comienza con los numeros 0 y 1 y apartir de estos , cada termino es la suma de los 2 anteriores. Es una relacion de recurrencia.

Ejemplo:0, 1,1,2,3,5,8,13,21,34,55, … :

1ª operación : 0+1=1 ; 2ª operación : 1+1=2 ; 2ª operación : 1+2=3 ; 4ª operación : 2+3=5 ……

EJERCICIO: hacer la serie de fibonacci hasta el termino n introducido por teclado.

Realizado con buble FOR:

```c
#include <stdio.h>

int main (){
    int x ,y , i, n, res;
    printf ("calcula serie de fibonacci hasta el numero  : ");
    scanf ("%i",&n);
    x = 0;
    y = 1;
    res = 0;

    for (i = 1; i<=n ; i++){
        res = x + y;
        printf ("%i, ",res);
        x = y;
        y = res;
    }

    system ("pause");
    return 0;
}
```

```
C:\Users\pc2\Documents\fibonacci.exe
calcula serie de fibonacci hasta el numero  : 4
1, 2, 3, 5, Presione una tecla para continuar . . .
```

Realizado con buble WHILE:

```c
#include <stdio.h>

int main (){
    int x ,y , i, n, res;
    printf ("calcula serie de fibonacci hasta el numero  : ");
    scanf ("%i",&n);
    i = 1;
    x = 0;
    y = 1;

    while (i<=n ){
        res = x + y;
        printf ("%i, ",res);
        x = y;
        y = res;
        i++;
    }

    system ("pause");
    return 0;
}
```

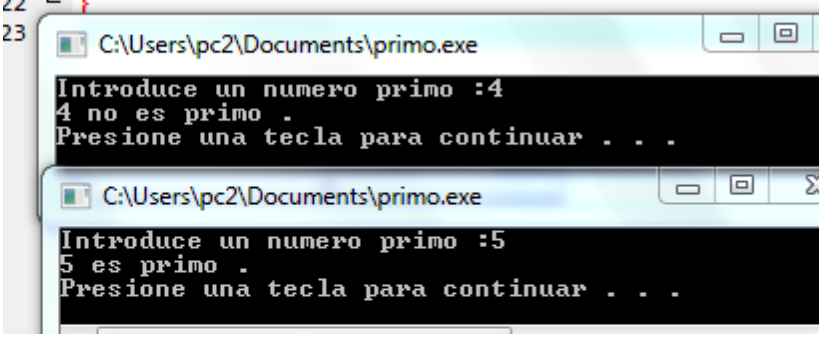
```
C:\Users\pc2\Documents\fibonacci.exe
calcula serie de fibonacci hasta el numero  : 4
1, 2, 3, 5, Presione una tecla para continuar . . .
```

Nos faltan el 0 y el 1 porque en la serie seria : 0, 1,1,2,3,5 y nuestro programa imprime 1,2,3,5 , bueno supongamos que queremos las siguientes operaciones apratir de las 2 primeras.

5) COMPROBAR SI UN NUMERO ES PRIMO :

Un numero es primo si no se le puede dividir por ningun numero exceptuando el 1 y si mismo.

```c
#include <stdio.h>
int main (){
    int x,i ,aux,j;  //  x para introduce un numero;
    // i es contador si aux distinto de 0 ;
    // aux es contenedor del resto;
    // j es contador si aux es igual a 0
    printf ("Introduce un numero primo :");
    scanf ("%i",&x);
    i=2; //empezamos en 2 porque el resto de x/1 da 0
    j = 0;
    aux = 0;
    while(i<x){
        aux = x % i;
        if ( aux == 0){j = 1;}
        i++;
    }
    if(j == 1){printf ("%i no es primo .\n",x);}
    else{printf ("%i es primo .\n",x);}

    system ("pause");
    return 0;
}
```



6) RUTAS DE ESCAPE :

Una ruta de escape es una forma de optimizar el funcionamiento de nuestro codigo en el bucle , es decir que el bucle siempre va a probar todos los casos del contador pero podemos facilitar el tiempo de ejecucion y cantidad de calculos en menos tiempo y claculos. Esto se consigue con CONDICIONES MULTIPLES que seran vias de escape.

OBS: cuantas menos veces se ejecute un bucle el programa obtendrá mayor eficiencia.

Para el ejemplo anterior su codigo no esta mal pero no es eficiente lo solucionamos asi:

`while(i<x && j != 1){`
`  aux = x % i;`
`  if ( aux == 0){j = 1;}`
`  i++;  }`
	Escribimos el mismo codigo, pero en el while ponemos en la condicion: i<x && j != 1 .
	Si j es distinto de 1 sigue el bucle pero si j = 1 no se cumple la condicion y salimos del bucles
	Por tanto se hacen menos calculos y es más eficiente.

EJERCICIO DE RUTAS DE ESCAPE: Imprime por pantalla todos los numeros primos entre 2 y 20.

```
1  //Imprime por pantalla todos los numeros primos entre 2 y 20.
2  #include <stdio.h>
3  int main (){
4      int i ,aux,j,k;
5      for (i = 2 ; i<20 ; i++){
6          k = 0;
7          for (j = 2 ; j < i && k != 1 ; j++){
8              aux = i % j;
9              if (aux == 0){
10                 k = 1;
11             }
12         }
13         if (k != 1){printf ("%i, ",i);}
14     }
15     system ("pause");
16     return 0;
17 }
18
```

```
C:\Users\pc2\Documents\...
2, 3, 5, 7, 11, 13, 17, 19, Pres
```

PROGRAMACION EN C: BLOQUE 4

1) **¿QUE ES UNA FUNCIÓN? :**

Funcion : lo primero , el int main es la funcion principal que se encarga de relaizar varios procesos que le pidamos.

Las funciones llevan a cabo procesos del programa (metodos) y son declaradas fuera del main para no generar tanto codigo dentro de este y generalizar casos ; es decir facilitarnos la vida .

Ahora para ver un ejemplo sencillo ecribire los pasos con 1º , 2º , 3º ….. que aparecerá en los comentarios del codigo asi que estaros atentos y atentas ;) .

1º HACER EL PROTOTIPO (antes del int main ponemos el mismo nombre y en vez de corchetes ponemos punto y coma.

2º llamamos a la funcion dentro del int main (con esto el programa entra dentro del int main y cuando vea una funcion se dirigirá a la función .

3º hará lo que tenga dicha funcion y al terminar se saldrá y volvera a la linea de abajo del main.

4º ejecuta la siente linea del main.

```
1   // FUNCIONES :
2
3   #include <stdio.h>
4
5   int maximo (); // ------1º Esto es el prototipo
6
7
8   int main (){
9
10      maximo(); // -----2º llamamos a la funcion dentro del main
11
12      printf ("Estamos en el main.\n\n"); // -----4º siguente ejecucion
13
14      system ("pause");
15      return 0;
16  }
17
18  int maximo (){   // ----3º entramos en la funcion con su proceso
19
20      printf ("estamos en la funcion.\n\n");
21      return 0;
22  }
```

```
C:\Users\pc2\Documents\primeraFuncion.exe
estamos en la funcion.

Estamos en el main.

Presione una tecla para continuar . . .
```

Avancemos un poco imprimamos los valores de x e y con una funcion:

Dentro de cada funcion (dominio) la variables que declaremos dentro de una funcion se quedan dentro y por tanto otra funciones no pueden usarlas (si declaramos x e y en main x e y no existen en la funcion)

Por tanto en la llamada del main ponemos el nombre de las variables y en la funcion ponemos otreo nombre pero del mismo tipo

Esto igual no lo entendeis ahora asi que pongo un codigo para que lo entendais

```
#include <stdio.h>

int maximo ();


int main (){

    int x , y ;

    x = 3;
    y = 10;

    maximo(x,y); // ponemos el nombre de las variables en el main


    system ("pause");
    return 0;
}

int maximo (int a , int b){  // en la funcion ponemos otros nombre a variables
// del mismo tipo ya que x es int e y es int

    printf ("El valor de x es %i y el valor de y es %i.\n\n",a,b);
    // llamamos a y b que almacenaran el valor de x e y es decir :
    // el valor de (a es x , osea 3 y el valor de b es y , osea 10)
    return 0;
}
```

```
C:\Users\pc2\Documents\primeraFuncion.exe
El valor de x es 3 y el valor de y es 10.

Presione una tecla para continuar . . .
```

Hasta aquí esto se llama paso de parametros . y basicamente dice que a =x y b = y

Siguiente ejemplo: Calculemos el valor mayor entre el x y el y :

```c
#include <stdio.h>
int maximo ();

int main (){

    int x , y , max; // declamos una variable max para el maximo

    x = 3;
    y = 10;

    max = maximo(x,y); // decimos que max almacena el resultado de la funcion maximo

    printf (" el valor maximo es : %i.\n\n", max);

    system ("pause");
    return 0;
}

int maximo (int a , int b){   // ahora aqui compararemos como siempre hemos hecho

    int aux; // va a ser donde almacenaremos el maximo

    if (a > b){
        aux = a;
    }
    else {
        aux = b;
    }

    return aux ; // NUEVO   DEVOLVEMOS AUX QUE ES EL MAXIMO.
}
```

```
C:\Users\pc2\Documents\primeraFuncion.exe
 el valor maximo es : 10.

Presione una tecla para continuar . . .
```

Hasta ahora en los RETURN siempre poniamos 0 , pues en las funciones se pone lo que va a devolver en este caso devuelve aux que almacena el valor maximo.

Otro dato importante es que en una funcion cuando devolvemos valores con return solo podemos devorver UN SOLO VALOR SOLO UNO.

Si no os habeis enterado muy bien aquí , tranquilos vamos a hacer más ejemplos , esto es una parte más compleja no hos alarmeis , al igual que avanzaremos y descubriremos cosas nuevas de las funciones .

A CONTINUACION HAREMOS MÁS EJEMPLOS:

2) Ejemplos con funciones :

Ejemplo 1 : calcular el factorial de dos numeros con una funcion :

```c
#include <stdio.h>
int faactorial (int a);  //  pongo el prototipo de la funcion factorial
int main (){

    int x , y , fact1 , fact2;
    printf (" introduce el primer numero para calcular su factorial : ");
    scanf ("%i",&x);
    printf (" introduce el segundo numero para calcular su factorial : ");
    scanf ("%i",&y);
    // declaramnos 4 enteros x e y para poner por teclado los numeros de los que queremos
    // conocer su factorial , fact1 y fact2  para recoger el resultado de la funcion de x e y
    fact1 = faactorial( x ); //  llamamos a la funcion factorial y la asignamos a fact1 para
    // recoger el valor por que factorial tiene un return que devuelve un valor
    fact2 = faactorial( y );

    printf ("el resultado de el factorial de x es %i.\n\n", fact1);
    printf ("el resultado de el factorial de y es %i.\n\n", fact2);

    system ("pause");
    return 0;
}
int faactorial ( int a ){  // realizamos la funcion factorial

    int i ,aux ;
    aux = 1;

    for (i = 1 ; i<= a; i++ ){
        aux = aux * i;
    }
    return aux ;
}
```

```
C:\Users\pc2\Documents\funcionFactorial.exe
 introduce el primer numero para calcular su factorial : 5
 introduce el segundo numero para calcular su factorial : 9
el resultado de el factorial de x es 120.

el resultado de el factorial de y es 362880.

Presione una tecla para continuar . . .
```

Con esto devemos fijarnos en una cosa y es que con una sola funcion hemos calculado 2 cosas es decir hemos generalizado el codigo para usa la funcion mas de una vez lo que apartir de ahora haremos muy amenudo para escribir menos y hacer mas eficiente nuestro codigo.

Ejemplo 2 : ver si 2 numeros diferentes son primos o no con una funcion :

```c
#include <stdio.h>
int primo(int a);
int main (){
    int x , y , prim1 , prim2;

    printf ("introduce el primer numero : ");
    scanf ("%i",&x);
    printf ("introduce el segundo numero : ");
    scanf ("%i",&y);

    prim1 = primo(x);
    prim2 = primo(y);

    if (prim1 == 0){
        printf ( "el nuemro %i es primo, \n\n",x);
    }
    else {  printf ( "el nuemro %i NO es primo, \n\n",x);}
    if (prim2 == 0){
        printf ( "el nuemro %i es primo, \n\n",y);
    }
    else {  printf ( "el nuemro %i NO es primo, \n\n",y);    }
    system ("pause");
    return 0;
}
int primo ( int a ){
    int i,j,aux;
    j = 0;
    // usaremos j como ruta de escape en el momento que no sea 0
    for ( i = 2; i < a && j ==0; i++){
        aux = a % i;
        if( aux == 0){j = 1;}
    }
    return j ;
}
```

```
introduce el primer numero : 5
introduce el segundo numero : 10
el nuemro 5 es primo,

el nuemro 10 NO es primo,

Presione una tecla para continuar . . .
```

3) Funciones de tipo void :

Las funciones void no devuelven nada , es decir no tienen return y por tanto no se puede almacenar en una variable , se utilizan para no hacer nada en el int main. Una vez llamadas en el main se encargan de hacer todo lo que ellas hagan el main SOLO LAS LLAMA .

```c
#include <stdio.h>

void factorial ();

int main (){

    factorial ();

    system ("pause");
    return 0;
}

void factorial (){
    int x,i,aux;

    aux = 1;

    printf ("introduce un numero entero: ");
    scanf ("%i",&x);

    for (i = 1; i <= x ; i++){
        aux = aux * i;
    }

    printf ("El factorial del numero %i es %i.\n\n",x,aux);

}
```

```
C:\Users\pc2\Documents\voidfactorial.exe
introduce un numero entero: 5
El factorial del numero 5 es 120.

Presione una tecla para continuar . . .
```

4) PUNTEROS Y PASOS POR REFERENCIA:

Cuando utilizamos “&” nos referimos a un referencia .

Los punteros solo señalan un valor de una direccion dada , se usan para el paso de parametros por referencias de funciones, se usa el simpolo “ * ” para indicar el valor que hay en una direccion.

Basicamente los punteros señalan y los & son la direccion de memoria

```c
#include <stdio.h>



int main (){

    int x , *y; // declarmos x  y declaramos y como un puntero

    x = 4; // x vale 4
    y =&x; // y toma como referencia de memoria el valor de x

    // LO QUE OCURRE ES QUE Y APUNTA AL VALO DE X (osea 4)
    printf ("el valor de *y es %i.\n\n",*y);

    system ("pause");
    return 0;
}
```

```
C:\Users\pc2\Documents\puntero.exe
el valor de *y es 4.

Presione una tecla para continuar . . .
```

Ejemplo : metodo de referencia

Consiste en intercambiar el valor de dos variables mediante una funcion:

```c
#include <stdio.h>
void cambio (int *a , int *b , int c);
int main (){

    int x , y , z; // declarmos x  , y  ,z

    x = 5; // x vale 5
    y = 10; // y vale 10
    z = 20;

    cambio (&x,&y, z); // funcion cambio que recive las direcciones de memoria
    printf ("el valor de de x es %i, y el valor de y es %i , el valor de z es %i.\n\n",x,y,z);

    system ("pause");
    return 0;
}
void cambio (int *a , int*b , int c){// funcion void porque no devuelve nada
// que recive punteros
int aux;

aux = *a;
*a =*b;
*b=aux ;

c = 100;
}
```

```
C:\Users\pc2\Documents\puntero.exe
el valor de de x es 10, y el valor de y es 5 , el valor de z es 20.
Presione una tecla para continuar . . .
```

El valor de x ahora vale lo que le hemos dado a y y viceversa para y , pero z no ha cambiado sigue siendo 20 esto es porque no se le a asignado ninguna teferencia de memoria a c por tanto nop cambia …

PROGRAMACION EN C: BLOQUE 5

1) ARRAYS O VECTORES :

Un array es una estructura que almacena dentro de ella varias variables del mismo tipo.

Se declaran tipo nombre [ ] = { _ _ _ _.....}

Ejemplo:

Si declamos lo siguiente : int array[ ] = { 1 ,3 ,5 ,4 ,7 ,9 ,5}; la forma que la maquina ve esto es la siguiente:

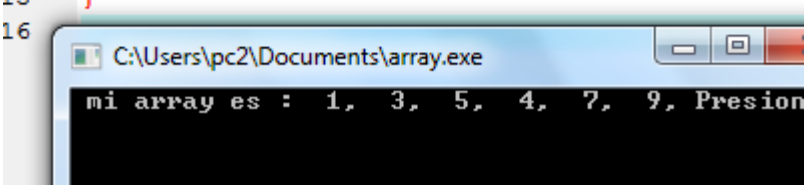
Valores:	1	3	5	4	7	9	5
Posiciones :	0	1	2	3	4	5	6

La maquina empieza en la posicion 0 que para nosotros seria la posicion 1 , esto es asi porque si.

Un dato muy curioso es que los vectore no necesitan llamarse como referencias de memoria porque ya lo son en si.

Hagamos un codigo que imprima este array o vector:

```
1   #include <stdio.h>
2
3   int main (){
4
5       int array[] = {1,3,5,4,7,9,5};
6
7       int i;
8       printf (" mi array es : ");
9       for (i = 0 ; i <= 5 ; i++ ){
10          printf (" %i, ",array[i]);
11      }
12
13      system ("pause");
14      return 0;
15  }
16
```



Otro ejemplo:

Dar un valor a un vector de 10 enteros y leerlos después:

```c
#include <stdio.h>

int main (){

    int arr[10]; // de esta forma digo que el arr tiene 10 elementos
    int i;


    for (i = 0 ; i <10 ; i++ ){
        printf (" dale valor al vector en la posicion %i: ",i);
        scanf ("%i",&arr[i]);
    }
    printf (" mi array es : ");
    for (i = 0 ; i <10 ; i++ ){
        printf (" %i, ",arr[i]);
    }

    system ("pause");
    return 0;
}
```

```
C:\Users\pc2\Documents\array.exe
dale valor al vector en la posicion 0: 1
dale valor al vector en la posicion 1: 2
dale valor al vector en la posicion 2: 3
dale valor al vector en la posicion 3: 4
dale valor al vector en la posicion 4: 5
dale valor al vector en la posicion 5: 6
dale valor al vector en la posicion 6: 7
dale valor al vector en la posicion 7: 8
dale valor al vector en la posicion 8: 9
dale valor al vector en la posicion 9: 10
mi array es :  1,  2,  3,  4,  5,  6,  7,  8,  9,  10, Presione una te
continuar . . .
```

Ahora vamos utilizar un vector en una funcion:

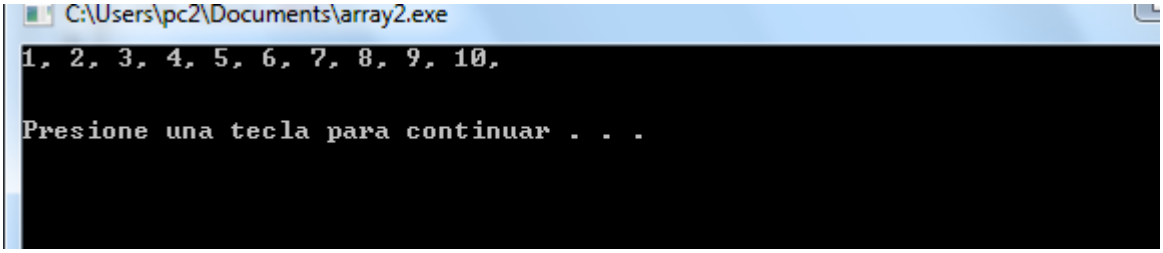
Introducir elementos a un vector mediante una función:

```c
#include <stdio.h>
void fun ( int a[]);    // prototipos
void imprimir (int x[]);
int main (){

    int arr[2]; // 1º declaramos el vector de 2 elementos por ejemplo

    fun (arr); // 2º llamada a la funcion fun (con poner el nombre del vector vale)
    // no es necesatio corchetes

    imprimir (arr); // funcion que imprimira los numeros por pantalla

    system ("pause");
    return 0;
}
void fun ( int a[]){ // funcion para introducir datos
    int i;
    for (i = 0;i < 2; i++){
        printf ("introduce un valor para el vector: ");
        scanf ("%i",&a[i]);
    }
}
void imprimir (int x[]){
    int i;
    for (i = 0;i < 2; i++){
        printf ("%i, ",x[i]);
    }
    printf ("\n\n\n");
}
```

```
C:\Users\pc2\Documents\array.exe
introduce un valor para el vector: 5
introduce un valor para el vector: 8
5, 8,


Presione una tecla para continuar . . .
```

Ejemplo: ordenar un vector de 10 enteros de menor a mayor mediante una funcion :

```
2   void ordenar (int a[]);     // prototipos
3   void imprimir (int x[]);
4   int main (){
5
6       int v[10] = {7,4,6,8,10,5,3,2,1,9};
7
8       ordenar (v);
9       imprimir (v);
10
11      system ("pause");
12      return 0;
13  }
14  void ordenar ( int a[]){
15      int i,j,aux;
16      for (i = 0;i < 10; i++){
17          for (j = i;j < 10; j++){
18              if (a[i] > a[j]){
19                  aux = a[i];
20                  a[i] = a[j];
21                  a[j] = aux;
22              }
23          }
24      }
25  }
26  void imprimir (int x[]){
27      int i;
28      for (i = 0;i < 10; i++){
29          printf ("%i, ",x[i]);
30      }
31      printf ("\n\n\n");
32  }
```



2) Array con punteros:

```c
#include<stdio.h>

int main(){

    int i;
    int vect[] = {1,2,3};

    for (i=0; i<3; i++){
        printf ("%i",*(vect+i));
        // CON ESTO (*(nombre+posicion)) accedemos a que imprima lo que contiene
        // la posicion ; en este caso imprime todos los valores de i osea 1 2 y 3
        // ejemplo vect[0] osea 1 en nuestro array se pone *(vect+0)
        // EN ESTE CASO QUEREMOS QUE IMPRIMA TODOS ASIQUE *(vect+i).
    }
    system ("pause");
    return 0;
}
```

```
C:\Users\pc2\Documents\arrayPunteros1.exe
123Presione una tecla para continuar . . .
```

Ejemplo: introducir valores numericos en un array mediante punteros.

```c
#include<stdio.h>
void funcion (int *vec);
void leer (int *vec);
int main(){

    int i;
    int vec[3];

    funcion (vec);
    leer (vec);

    system ("pause");
    return 0;
}

void funcion (int *vec){
    int i;
    for (i=0;i<3;i++){
        printf ("introduce un valor para la posicion %i:",i+1);
        scanf ("%i", (vec+i));
    }
}
void leer (int *vec){
    int i;
    for (i=0;i<3;i++){
        printf ("%i ",*(vec+i));

    }
}
```

```
C:\Users\pc2\Documents\arrayPunteros1.exe
introduce un valor para la posicion 1:8
introduce un valor para la posicion 2:4
introduce un valor para la posicion 3:3
8 4 3 Presione una tecla para continuar . . .
```

Ejemplo: calcula las probabilidades de que salga cada uno de los numeros de un dado con un vectror usando punteros , calcular con 10000 iteraciones.

```c
#include<stdio.h>
void leer (int*vec);
void sorteo (int *vec);

int main(){

    int vec[] = {0,0,0,0,0,0}; // este vector va ser un contador
    // ejemplo si sale un 1 y un 6 este vector quedara {1,0,0,0,0,1}

    sorteo (vec);
    leer (vec);
    system ("pause");
    return 0;
}
void sorteo (int *vec){
    srand (time(NULL)); // esto es para que salgan numeros aletorios
    // no es muy importante , solo para este programa
    int i, aux;
    for(i=0;i<10000;i++){
        aux = rand()%6; // aux va a recibir 6 numeros
        // si pusiera rand()%4 reciviria 4 ...
        switch(aux){
            case 0:{
            *(vec+aux) += 1;
                break;
            }
            case 1:{
                *(vec+aux) +=1;
                break;
            }
            case 2:{
                *(vec+aux) +=1;
                break;
            }
            case 3:{
                *(vec+aux) +=1;
                break;
            }
            case 4:{
                *(vec+aux) +=1;
                break;
            }
            case 5:{
                *(vec+aux) +=1;
                break;
            }
        }
    }
}
void leer (int*vec){
    int i;
    float x;

    for (i=0;i<6;i++){
        x=(*(vec+i)*100)/10000.; // ponemos punto porque devolvera flotante
        printf ("la probabilidad de que el numero %i salga es de: %f.\n",i+1,x);
    }
}
```

```
C:\Users\pc2\Documents\arrayPunteros1.exe
la probabilidad de que el numero 1 salga es de: 16.290001.
la probabilidad de que el numero 2 salga es de: 17.000000.
la probabilidad de que el numero 3 salga es de: 16.590000.
la probabilidad de que el numero 4 salga es de: 17.160000.
la probabilidad de que el numero 5 salga es de: 16.950001.
la probabilidad de que el numero 6 salga es de: 16.010000.
Presione una tecla para continuar . . .
```

3) Matrices :

Una matriz es un conjunto de vectore o un vector de varias dimensiones que es lo mismo.

Una matriz siempre se va a construir mediante dos bucles , uno para filas y otro para columnas

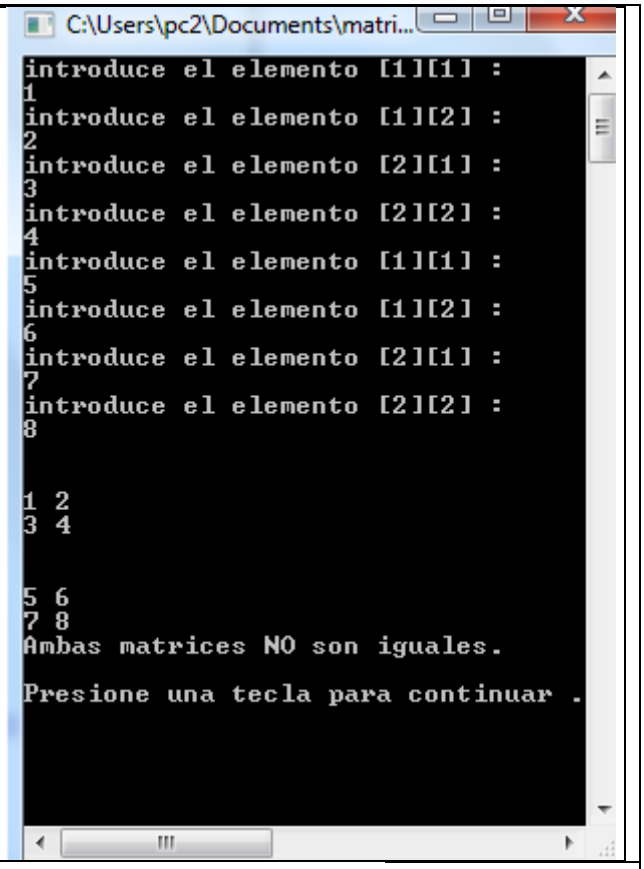
EJEMPLO: CONSTRUIR UNA MATRIZ Y LEERLA POR PANTALLA

```c
#include<stdio.h>
void introduce (int matriz[][3]);
void leer (int matriz[][3]);
int main(){

    int matriz[3][3]={{0}} ;// esto es un vector de 2 dimensiones
    // esta será una matriz 3 x 3

    introduce (matriz);
    leer (matriz);
    system ("pause");
    return 0;
}
void introduce (int matriz[][3]){ // siempre que hagamos una matriz es
//OBLIGATORIO poner el numero de elementos que tendrán las columnas en este caso 3
    int i , j;
    // matriz
    for (i=0;i<3;i++){
        for (j=0;j<3;j++){
            printf ("introduce el elemento [%i][%i] :\n",i+1,j+1);
        scanf ("%i", &matriz[i][j]);
        }
    }
}
void leer (int matriz[][3]){
    int i,j;
    for (i =0;i<3;i++){
        for(j=0;j<3;j++){
            printf("%i ",matriz[i][j]);
        }
        printf ("\n");
    }
}
```

```
C:\Users\pc2\Documents\matriz.exe
introduce el elemento [1][1] :
1
introduce el elemento [1][2] :
2
introduce el elemento [1][3] :
3
introduce el elemento [2][1] :
4
introduce el elemento [2][2] :
5
introduce el elemento [2][3] :
6
introduce el elemento [3][1] :
7
introduce el elemento [3][2] :
8
introduce el elemento [3][3] :
9
1 2 3
4 5 6
7 8 9
Presione una tecla para continuar . . .
```

Otro elemplo : Comparar 2 matrizes:

```c
#include<stdio.h>
void introduce (int m[][2]);
void compara (int m1[][2] ,int m2[][2]);
void leer (int m[][2]);
int main(){

    int m1[2][2] ;
    int m2[2][2] ;

    introduce (m1);
    introduce (m2);
    printf ("\n\n");
    leer (m1);
    printf ("\n\n");
    leer (m2);

    compara(m1,m2);
    system ("pause");
    return 0;
}
void introduce (int m[][2]){
    int i , j;

    for (i=0;i<2;i++){
        for (j=0;j<2;j++){
            printf ("introduce el elemento [%i][%i] :\n",i+1,j+1);
        scanf ("%i", &m[i][j]);
        }
    }
}
```




```c
//la funcion compara aparte de 2 bucles necesita un condicional donde m1 sea distinto de m2
// y cuando sea distinto tengo variable aux que valdrá 1 y la inicio a 0
// esto lo que hace es cuando una matriz tenga cualquier elemento diferente aux le va a dar
// el numero 1 y antes era 0 , pues en cuanto un solo elemento sea distinto tiene que
// dejar de comparar , esto se hace poniendo una ruta de escape en cada uno de los bucles
// donde aux = 0 y solamente dejara de ser 0 cuando 2 elementos sean diferentes.
void compara (int m1[][2] ,int m2[][2]){
    int i,j,aux;

    aux =0;
    for (i =0;i<2 && aux== 0;i++){
        for(j=0;j<2 && aux== 0;j++){
            if (m1[i][j] != m2[i][j]){
                aux = 1;
            }
        }
    }
    if (aux == 0){
        printf ("Ambas matrices son iguales. \n\n");
    }
    else{
        printf ("Ambas matrices NO son iguales. \n\n");
    }
}
void leer (int m[][2]){
    int i,j;
    for (i =0;i<2;i++){
        for(j=0;j<2;j++){
            printf("%i ",m[i][j]);
        }
        printf ("\n");
    }
}
```

4) **Strings y sus comparaciones:**

Un String es la forma que tenemos en informatica de almacenar una frase , esto esta muy relacionado con los vectores a la hora de almacenar.

Un string se declara asi :

char nombre[nº_elementos+1];	
char frase[13];	**Hola que tal**

En este ejemplo vemos que la frase " Hola que tal " tiene 12 elementos la suma de letras y espacios .

¿Por qué se pone 13? Bueno esto es devido a que al guardar una cadena de caracteres siempre se declara un elemento mas esto se debe a que todos los strings termian con un carácter que es \0 (contrabarra cero) es asi porque ese carácter indica donde termina el String.

Por tanto la frase a almacenar seria : Hola que tal\0

Para introducir una frase del tamaño deseado usaremos :

fgets(nombre,nº_elementos+1,stdin); hay mas fromas pero esta el la mejor .

stdin es para introducir por teclado.

Ejemplo : introducir una frase como cadena de caracteres(strings) con un tamaño de 12 elementos:

```
1   #include<stdio.h>
2
3   int main(){
4
5       char frase[13]; // 12 elementos + 1 ( lo que he contado)
6
7       printf ("Introduce una frase: ");
8       fgets(frase,13,stdin);
9
10      printf ("%s\n\n",frase);
11      system ("pause");
12      return 0;
13  }
14
```

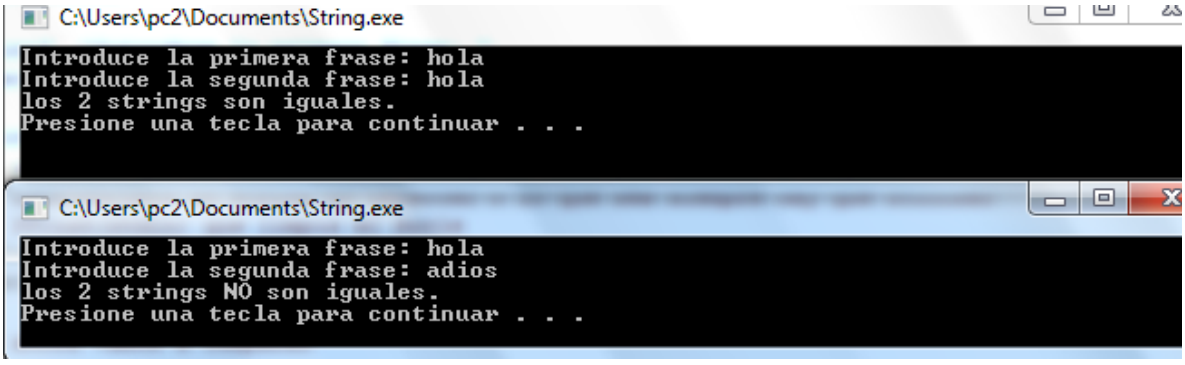
```
C:\Users\pc2\Documents\String.exe
Introduce una frase: Hola que tal estamos
Hola que tal

Presione una tecla para continuar . . .
```

Aquí seguramente te preguntaras : si has puesto la frase " Hola que tal estamos "
¿por que solo aparece Hola que tal ?

Bueno , lo he hecho aposta ya que asi os demuestro que solo se imprimira 12 caracteres mas el \0 (osea 13)
y si os fijais Hola que tal ya tiene 12 mas el \0 (que ahora no veis por ninguna parte , solo os lo cuento) tiene 13
por tanto todo lo que pongamos pasados 13 caracteres no se almacenará en memoria.

Ejemplo : comparar 2 cadenas de cracteres :

```c
#include<stdio.h>
#include<string.h> // ponemos la libreria de strings
int main(){

    char frase1[5];
    char frase2[5];

    printf ("Introduce la primera frase: ");
    fgets(frase1,5,stdin);

    printf ("Introduce la segunda frase: ");
    // recordad que si introducimos cualquier cosa por teclado y despues
    // introducimos un string un caracter o lo que sea siempre hay que utilizar
    // fflush(stdin) que limpia el bucle
    fflush(stdin);
    fgets(frase2,5,stdin);

    // ahora vamos a comparar

    if (strcmp(frase1,frase2) == 0){// strcmp compara strings ; ES UNA FUNCION AUXILIAR
        printf ("los 2 strings son iguales.\n");
    }
    else{
        printf ("los 2 strings NO son iguales.\n");
    }

    system ("pause");
    return 0;
}
```



Mas adelante usaremos mas funciones auxiliares que nos serán de bastante ayuda.

Ejemplo : como acceder a un carácter dentro de string :

Antes de comenzar voy revelaros un engaño que he hecho y eso es que el "fgets()" al introducir una frase va a terminar siempre con \n y nosotros queremos que termine con \0 esto es un pequeño inconveniente pero de facil solucion ya que en este ejercicio aprenderemos como cambiar elementos de un string.

FORMA CORRECTA

```c
#include<stdio.h>
#include<string.h>
int main(){

    char c1[10];

    printf("Introduzca una frase: ");
    fgets(c1,10,stdin);

    cambio(c1);

    printf("%s\n",c1);

    system ("pause");
    return 0;
}
void cambio(char c1[]){
    int i;
    for(i=0;i<10;i++){
        if (c1[i] == '\n'){
            c1[i] = '\0';
        }
    }
}
```

```
C:\Users\pc2\Documents\accederString.exe
Introduzca una frase: hola
hola
Presione una tecla para continuar . . .
```

FORMA ERRONEA

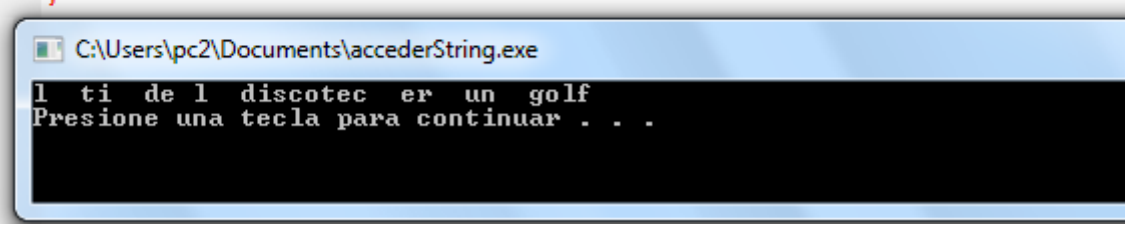
```c
#include<stdio.h>
#include<string.h>
int main(){

    char c1[10];

    printf("Introduzca una frase: ");
    fgets(c1,10,stdin);


    printf("%s\n",c1);

    system ("pause");
    return 0;
}
```

```
C:\Users\pc2\Documents\accederString.exe
Introduzca una frase: hola
hola

Presione una tecla para continuar . . .
```

Efectivamnte en la forma erronea vemos como al ejecutar se produce un salto de linea \n y en la forma correcta despues del hola continua con Presione una tecla

Ejemplo: cambiar en una frase dada la letra a por un espacio:

```c
#include<stdio.h>
#include<string.h>
 void funcion (char c1[]);
int main(){
    // creamos nuestro string y lo inicializamos
    char c1[] = "la tia de la discoteca era una golfa";

    funcion(c1);

    printf("%s\n",c1);

    system ("pause");
    return 0;
}
void funcion (char c1[]){
    int limite,i;

    limite = strlen(c1);// devuelve como de larga es una cadena
    for(i=0;i<limite;i++){
        if (c1[i] == 'a'){
            c1[i] = ' ';
        }
    }
}
```



Como vemos efectivamente ha cambiado todas las "a" por espacios

Recordemos para meter datos usamos el fgets()

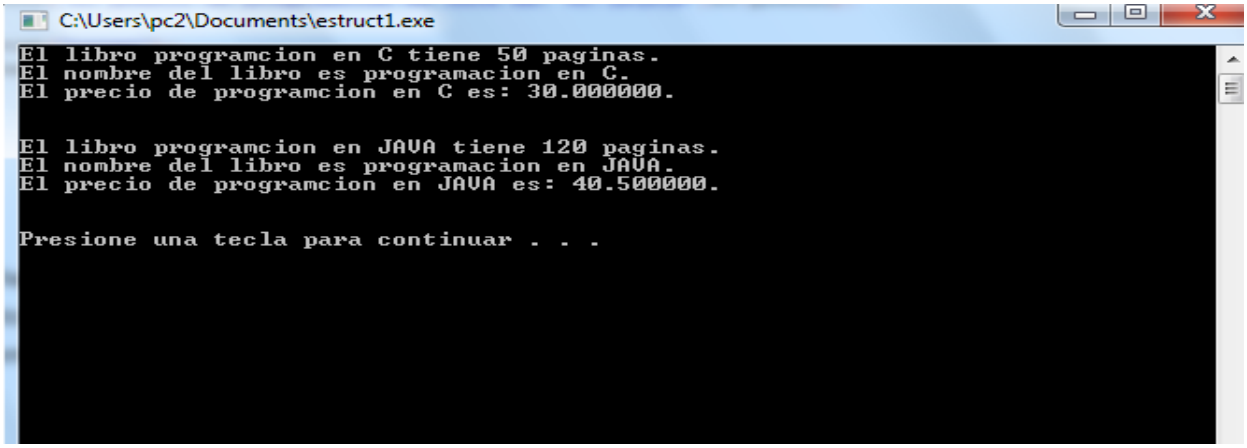
PROGRAMACION EN C: BLOQUE 6

¿Qué son las estructuras ? :

- Una estructura es declarada fuera del main y sirve para agrupar muchas variables , siendo estas variables de muchos tipos .
- Basicamente la estructura al ser llamada otorga la posibilidad de acceder a cualquiera de nuestras variables , la estructura es un tipo , con esto me refiero a que es lo mismo que (int , char , float …)
- Una estructura al ser un nuevo tipo , es un tipo creado por nosotros con los tipos que queramos :
- Ejemplo : un libro tiene : paginas , nombre y precio.
- Accedemos a ellas como NOMBRE_VARIABLE.paginas , strcpy(nombreVar.nombre) para string ….

Veamos un ejemplo sencillo:

```c
#include <stdio.h>
#include<string.h>
// estructura :
typedef struct{
    int paginas;
    char nombre[50];
    float precio;
}libro;//--------> LIBRO COMO NOMBRE DE LA ESTRUCTURA

int main(){
    libro PC,PJ; // creo dos variables de tipo libro que contiene  paginas , nombre , precio
     // accedo a las variables del tipo libro de la siguiente forma :

    PC.paginas = 50; // la variable paginas de PC es de 50
    strcpy(PC.nombre,"programacion en C"); // escribimos el nombre de nuestro libro
    PC.precio = 30; // asignamos un precio

    PJ.paginas = 120;
    strcpy(PJ.nombre,"programacion en JAVA");
    PJ.precio = 40.5;

    printf("El libro programcion en C tiene %i paginas.\n",PC.paginas);
    printf("El nombre del libro es %s.\n",PC.nombre);
    printf("El precio de programcion en C es: %f.\n\n\n",PC.precio);

    printf("El libro programcion en JAVA tiene %i paginas.\n",PJ.paginas);
    printf("El nombre del libro es %s.\n",PJ.nombre);
    printf("El precio de programcion en JAVA es: %f.\n\n\n",PJ.precio);

    system("pause");
    return 0;
}
```



Pero ahora hagámoslo más genérico es decir imaginemos que tenemos 10 libros , es muy pesado crear todos uno a uno por tanto , la potencia de las estructuras va a ser que podemos crear vectores de nuestra estructura como en el siguiente ejemplo:

```c
// queremos ver el precio de los 10 capitulos del libro programacion en C
#include <stdio.h>
#include<string.h>
// estructura :
typedef struct{
    int paginas;
    char nombre[50];
    float precio;
}libro;//--------> LIBRO COMO NOMBRE DE LA ESTRUCTURA

int main(){
    int i;
    libro PC[10];// creamos 10 estructuras diferentes
    // accedemos a casa una de ellas
    for (i = 0; i<10;i++){
        PC[i].precio = 25 + i; // asignamos precios,cada libro vale 1 unida mas que el anterior
        printf("El precio de capitulo %i de programacion en C es de: %f.\n",i+1,PC[i].precio);
    }
    system("pause");
    return 0;
}
```

```
C:\Users\pc2\Documents\estruct2.exe
El precio de capitulo 1 de programacion en C es de: 25.000000.
El precio de capitulo 2 de programacion en C es de: 26.000000.
El precio de capitulo 3 de programacion en C es de: 27.000000.
El precio de capitulo 4 de programacion en C es de: 28.000000.
El precio de capitulo 5 de programacion en C es de: 29.000000.
El precio de capitulo 6 de programacion en C es de: 30.000000.
El precio de capitulo 7 de programacion en C es de: 31.000000.
El precio de capitulo 8 de programacion en C es de: 32.000000.
El precio de capitulo 9 de programacion en C es de: 33.000000.
El precio de capitulo 10 de programacion en C es de: 34.000000.
Presione una tecla para continuar . . .
```

¿COMO SE DECLARA UNA ESTRUCTURA?

typedef struct{
variable1;
variable2;
…….
variableN;
}nombre_estructura;

Ejemplo: crear un formulario:

```c
#include<stdio.h>

typedef struct{
    char nombre[50]; // string nombre
    char empleo[50]; // string empleo
    int edad;
}registro;

int main(){
    registro f1,f2;

    printf("Introduce valores para el formulario 1: \n\n");
    printf("Introduce 1 nombre: ");
    fgets(f1.nombre,50,stdin); // para introducir un string uso fgets
    printf("Introduce 1 empleo: ");
    fflush(stdin); // ponemos fflush para limpiar el buffer ya que es el 2° string
    fgets(f1.empleo,50,stdin);
    printf("Introduce 1 edad: ");
    scanf("%i",&f1.edad);

    printf("Introduce valores para el formulario 2: \n\n");
    printf("Introduce 2 nombre: ");
    fflush(stdin);
    fgets(f2.nombre,50,stdin); // para introducir un string uso fgets
    printf("Introduce 2 empleo: ");
    fflush(stdin); // ponemos fflush para limpiar el buffer ya que es el 2° string
    fgets(f2.empleo,50,stdin);
    printf("Introduce 2 edad: ");
    scanf("%i",&f2.edad);

    printf("los valores de los nombre son: \n");
    printf("el nombre de f1 es: %s",f1.nombre);
    printf("el nombre de f2 es: %s",f2.nombre);

    printf("el empleo de f1 es: %s",f1.empleo);
    printf("el empleo de f2 es: %s",f2.empleo);

    printf("la edad de f1 es: %i\n",f1.edad);
    printf("la edad de f2 es: %i\n",f2.edad);


    system("pause");
    return 0;
}
```

```
C:\Users\pc2\Documents\formulario.exe
Introduce valores para el formulario 1:

Introduce 1 nombre: pepe
Introduce 1 empleo: butanero
Introduce 1 edad: 34
Introduce valores para el formulario 2:

Introduce 2 nombre: rosa
Introduce 2 empleo: zampabollos
Introduce 2 edad: 59
los valores de los nombre son:
el nombre de f1 es: pepe
el nombre de f2 es: rosa
el empleo de f1 es: butanero
el empleo de f2 es: zampabollos
la edad de f1 es: 34
la edad de f2 es: 59
Presione una tecla para continuar . . .

--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

EJEMPLO DE NIVEL :

Queremos hacer un programa traductor de inglés a español el cual nos permita crear traducciones y buscarlas

```c
// TRADUCTOR

#include<stdio.h>
#include<string.h>
#define N 50  // donde aparezca N se cambia por 50
// LA ESTRUCTURA VA A ALMACENAR 2 PALABRAS UNA EN ESPAÑOL Y OTRA EN INGLES
typedef struct{
    char eng[N];  // almacena palabras en ingles
    char esp[N];  // almacena palabras en español
    int lleno;
}traductor;

traductor trad[N]; // vector de 50 estructuras traductor con nombre trad

void vacio();
void anadir();
void traducir();
void tradu(int op);


int main(){
    int op;
    char key;

    vacio(); // en la variable lleno pone todas la variables lleno de todas las etructuras a 0

    do{
        do{
            printf("ELIGE QUE DESEAS HACER:\n\n");
            printf("(1) INSERTAR PALABRAS AL TRADUCTOR.\n");
            printf("(2) BUSCAR TRADUCION DE UNA PALABRA.\n");
            scanf("%i",&op);
        }
        while(op<1 || op>2);

        switch(op){
            case 1:
                anadir();
                break;
            case 2:
                traducir();
                break;
        }
        printf("Introduce si deseas realizar ptra operacion: (S/N)");
        scanf("%c",&key);
    }
    while(key == 'S' || key == 's');
    system("pause");
    return 0;
}

```

```c
void vacio(){ // todas la variables lleno de las estructuras se cambian a 0
    int i;
    for (i=0 ; i < N; i++){
        trad[i].lleno = 0;
    }
}
void anadir(){ // inrtroduce palabras en ingles con su traduccion al español
// solo se meten palabras si el lleno de una estructura esta a 0
    int i , aux ;
    aux =0;
    for (i =0 ; i<N && aux ==0 ; i++){
        if(trad[i].lleno == 0){
            printf("Introduce la palabra en Ingles: ");
            fflush(stdin);
            fgets(trad[i].eng,N,stdin);
            cambio(trad[i].eng);
            printf("Introduce la palabra en Espanol: ");
            fflush(stdin);
            fgets(trad[i].esp,N,stdin);
            cambio(trad[i].esp);
            trad[i].lleno =1; // para no sobrescribir la palabra
            aux = 1;
        }
    }
}
```

```c
void traducir(){
    int op;
    do{
        printf("ELIGE QUE DESEAS HACER:\n\n");
        printf("(1) TRADUCIR DE INGLES A ESPANOL.\n");
        printf("(2) TRADUCIR DE ESPANOL A INGLES.\n");
        scanf("%i",&op);
        }
        while(op<1 || op>2);

        switch(op){
            case 1:
                tradu(op);
                break;
            case 2:
                tradu(op);
                break;
        }
}
```

```c
void tradu(int op){ // busca si la palabra tiene traduccion dentro de todas las estructuras
    int i,j,temp;
    char aux[N];
    temp = 0;
    printf("introduce la palabra que deseas buscar: ");
    fflush(stdin);
    fgets(aux,N,stdin);
    cambio(aux);
    if (op == 1){
        for(i=0;i<N && temp ==0;i++){
            j= strcmp(aux,trad[i].eng);  // strcmp compara strings
            // si ambas palabras son iguales j =0 sino j distinto de 0
            if (j == 0){
                printf("La traduccion de %s es %s.\n",trad[i].eng,trad[i].esp);
                temp = 1;
            }
        }
    }
    else{
        for(i=0;i<N && temp == 0;i++){
            j= strcmp(aux,trad[i].esp);  // strcmp compara strings
            // si ambas palabras son iguales j =0 sino j distinto de 0
            if (j == 0){
                printf("La traduccion de %s es %s.\n",trad[i].esp,trad[i].eng);
                temp = 1;
            }
        }
    }
}
```

```c
void cambio(char palabra[N]){ // hace que no haga salto de linea el programa
int i;
    for (i=0 ; i < N; i++){
        if (palabra[i] == '\n'){
            palabra[i] = '\0';
        }
    }
}
```

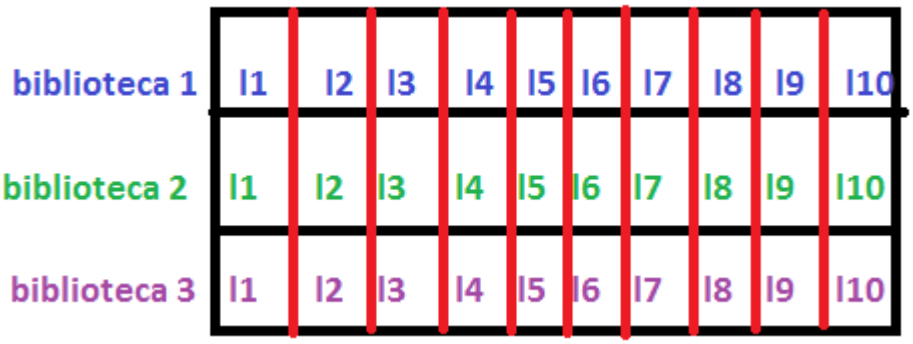
```
C:\Users\pc2\Documents\traductor.exe
ELIGE QUE DESEAS HACER:

(1) INSERTAR PALABRAS AL TRADUCTOR.
(2) BUSCAR TRADUCION DE UNA PALABRA.
1
Introduce la palabra en Ingles: go
Introduce la palabra en Espanol: ir
Introduce si deseas realizar ptra operacion: (S/N)s
ELIGE QUE DESEAS HACER:

(1) INSERTAR PALABRAS AL TRADUCTOR.
(2) BUSCAR TRADUCION DE UNA PALABRA.
2
ELIGE QUE DESEAS HACER:

(1) TRADUCIR DE INGLES A ESPANOL.
(2) TRADUCIR DE ESPANOL A INGLES.
1
introduce la palabra que deseas buscar: go
La traduccion de go es ir.
Introduce si deseas realizar ptra operacion: (S/N)n
Presione una tecla para continuar . . .
```

OTRO EJEMPLO DE NIVEL: (ESTRUCTURAS DENTRO DE ESTRUCTURAS)

GESTION DE BIBLIOTECAS (PARECIDO AL ANTERIOR EJERCICIO).

Cada vez que quiera crear un elemento de la realidad se utiliza una estructura , usando una estructura puedo crear muchos objetos del mismo tipo .

Podemos pensar también que una estructura dentro de otra estructura es una matriz de estructuras



Viene mas o menos a ser esta la idea pero no nos enrollemos mas y escribamos el código.

```c
// BIBLIOTECAS

#include<stdio.h>
#include<string.h>
#define N 50

typedef struct{
    char nombre_lib[N];  // nombre de un libro
    char autor[N];  // autor libro
    int lleno_lib;      // indica si la estructura esta llena
}libro;

typedef struct{
    char nombre_bi[N];  // nombre de la bibioteca
    libro lib[10];  // cada biblioteca tendrá 10 libros
    int lleno_bi;      // indica si la estructura esta llena
}biblioteca;

biblioteca bi[3]; // vector de 3 bibioteca es decir tendremos 30 libros

void vacio();
void cambio(char palabra[N]);
void anadebi();
void anadelib();
void consulta();

int main(){
    int op;
    char key;

    vacio(); // en la variable lleno pone todas la variables lleno de todas las etructuras a 0

```

```
33      do{
34          do{
35              printf("ELIGE QUE DESEAS HACER:\n");
36              printf("(1) anadir una nueva biblioteca.\n");
37              printf("(2) anadir un nuevo libro.\n");
38              printf("(3) consultar un libro.\n");
39              scanf("%i",&op);
40          }
41          while(op<1 || op>3);
42
43          switch(op){
44              case 1:{
45                  anadebi();
46                  break;
47              }
48              case 2:{
49                  anadelib();
50                  break;
51              }
52              case 3:{
53                  consulta();
54                  break;
55              }
56          }
57          printf("Introduce si deseas realizar otra operacion: (S/N): ");
58          fflush(stdin);
59          scanf("%c",&key);
60      }
61      while(key == 'S' || key == 's');
62      system("pause");
63      return 0;
64  }
```

```
65
66  void vacio(){ // todas la variables lleno de las estructuras se cambian a 0
67      int i,j;
68      for (i=0 ; i < 3; i++){
69          bi[i].lleno_bi = 0;
70          for (j = 0; j<10;j++){
71              bi[i].lib[j].lleno_lib = 0;
72          }
73      }
74  }
75  void cambio(char palabra[N]){ // hace que no haga salto de linea el programa
76  int i;
77      for (i=0 ; i < N; i++){
78          if (palabra[i] == '\n'){
79              palabra[i] = '\0';
80          }
81      }
82  }
83  void anadebi(){
84      int i , aux ;
85      aux =0;
86      for (i =0 ; i<3 && aux ==0 ; i++){
87          if(bi[i].lleno_bi == 0){
88              printf("Introduce un nombre para la biblioteca: ");
89              fflush(stdin);
90              fgets(bi[i].nombre_bi,N,stdin);
91              cambio(bi[i].nombre_bi);
92
93              bi[i].lleno_bi =1; // para no sobrescribir la palabra
94              aux = 1;
95          }
96      }
```

```
97      if(aux == 0){
98          printf("no queda ningun hueco libre para una nueva biblioteca.\n");
99      }
100 }
101 void anadelib(){
102     int i , op , aux;
103     aux = 0;
104     for (i =0 ; i<3 && aux == 0; i++){
105         if(bi[i].lleno_bi == 1){
106             printf("(%i) %s.\n",i,bi[i].nombre_bi);
107         }
108     }
109     scanf("%i",&op);
110         for (i =0 ; i<10 && aux == 0 ; i++){
111         if(bi[op].lib[i].lleno_lib == 0){
112             printf("Introduce el nombre del libro: ");
113             fflush(stdin);
114             fgets(bi[op].lib[i].nombre_lib, N, stdin);
115             cambio(bi[op].lib[i].nombre_lib);
116 
117             printf("Introduce el nombre del autor: ");
118             fflush(stdin);
119             fgets(bi[op].lib[i].autor,N,stdin);
120             cambio(bi[op].lib[i].autor);
121 
122             bi[op].lib[i].lleno_lib == 1;
123             aux =1;
124         }
125     }
126 }
127 void consulta(){
128     int op,i,j,aux;
```

```
129     char buscar[N];
130 
131     aux = 1;
132 
133     do{
134         printf("¿que quieres buscar, nombre del libro o nombre autor: \n\n");
135         printf("(1) Nombre Libro.\n");
136         printf("(2) Nombre Autor.\n");
137         scanf("%i",&op);
138     }
139     while(op<1 || op>2);
140 
141     switch(op){
142         case 1:{
143             printf("Introduce el nombre del libro: ");
144             fflush(stdin);
145             fgets(buscar, N, stdin);
146             cambio(buscar);
147 
148             for (i = 0 ; i<3; i++){
149                 aux = 1; // metemos un valor distinto por si los libros son = ya que aux antes siempre valia 0
150                 for (j =0 ; j<10; j++){
151                     aux = strcmp(buscar,bi[i].lib[j].nombre_lib);
152                     if(aux == 0){
153                         printf("El libro %s se encuentra en la biblioteca %s.\n\n",buscar,bi[i].nombre_bi);
154                         printf("El autor del libro es %s.\n\n",bi[i].lib[j].autor);
155                     }
156                 }
157             }
158             break;
159         }
160         case 2:{
```

```
161             printf("Introduce el nombre del autor: ");
162             fflush(stdin);
163             fgets(buscar,N,stdin);
164             cambio(buscar);
165
166             for (i = 0 ; i<3; i++){
167                 aux = 1; // metemos un valor distinto por si los libros son = ya que aux antes siempre valia 0
168                 for (j =0 ; j<10; j++){
169                     aux = strcmp(buscar,bi[i].lib[j].autor);
170                     if(aux == 0){
171                     printf("El libro %s esta en la biblioteca %s.\n\n",bi[i].lib[j].nombre_lib,bi[i].nombre_bi );
172                     }
173                 }
174             }
175             break;
176         }
177     }
178 }
```

```
C:\Users\pc2\Documents\bibliotecas.exe
ELIGE QUE DESEAS HACER:
(1) anadir una nueva biblioteca.
(2) anadir un nuevo libro.
(3) consultar un libro.
1
Introduce un nombre para la biblioteca: Burgos
Introduce si deseas realizar otra operacion: (S/N): S
ELIGE QUE DESEAS HACER:
(1) anadir una nueva biblioteca.
(2) anadir un nuevo libro.
(3) consultar un libro.
1
Introduce un nombre para la biblioteca: Madrid
Introduce si deseas realizar otra operacion: (S/N): s
ELIGE QUE DESEAS HACER:
(1) anadir una nueva biblioteca.
(2) anadir un nuevo libro.
(3) consultar un libro.
1
Introduce un nombre para la biblioteca: Cadiz
Introduce si deseas realizar otra operacion: (S/N): s
ELIGE QUE DESEAS HACER:
(1) anadir una nueva biblioteca.
(2) anadir un nuevo libro.
(3) consultar un libro.
1
no queda ningun hueco libre para una nueva biblioteca.
Introduce si deseas realizar otra operacion: (S/N): s
ELIGE QUE DESEAS HACER:
(1) anadir una nueva biblioteca.
(2) anadir un nuevo libro.
(3) consultar un libro.
2
(0) Burgos.
(1) Madrid.
(2) Cadiz.
0
Introduce el nombre del libro: programacion en C
Introduce el nombre del autor: alvar
Introduce si deseas realizar otra operacion: (S/N): S
```

```
ELIGE QUE DESEAS HACER:
(1) anadir una nueva biblioteca.
(2) anadir un nuevo libro.
(3) consultar un libro.
2
(0) Burgos.
(1) Madrid.
(2) Cadiz.
1
Introduce el nombre del libro: mi historia
Introduce el nombre del autor: chapas
Introduce si deseas realizar otra operacion: (S/N): S
ELIGE QUE DESEAS HACER:
(1) anadir una nueva biblioteca.
(2) anadir un nuevo libro.
(3) consultar un libro.
3
¿que quieres buscar, nombre del libro o nombre autor:

(1) Nombre Libro.
(2) Nombre Autor.
1
Introduce el nombre del libro: programacion en C
El libro programacion en C se encuentra en la biblioteca Burgos

El autor del libro es alvar .

Introduce si deseas realizar otra operacion: (S/N): s
ELIGE QUE DESEAS HACER:
(1) anadir una nueva biblioteca.
(2) anadir un nuevo libro.
(3) consultar un libro.
3
¿que quieres buscar, nombre del libro o nombre autor:

(1) Nombre Libro.
(2) Nombre Autor.
2
Introduce el nombre del autor: chapas
El libro mi historia se encuentra en la biblioteca Madrid.

Introduce si deseas realizar otra operacion: (S/N): N
Presione una tecla para continuar . . .
```

PUNTEROS A ESTRUCTURAS Y FUNCIONES CON ESTRUCTURAS:

Hasta ahora cuando declaramos algo fuera de cualquier función significa que estamos haciendo una variable de tipo global y por eso nosotros podemos utilizar las estructuras en cualquier tipo de función sin hacer ningún tipo de paso por referencia ni valor

Pero podemos crear la variable de la estructura dentro de una función , solo existirá esa variable en el ámbito de esa función , es decir solo existe esa variable en el territorio de esa función .

Para que exista haremos un paso por valor en una función que creemos.

Básicamente este concepto de punteros en estructuras no es difícil , trata de donde hay un punto de unión por una flecha .

Vamos a verlo con el siguiente ejemplo:

Realizar un programa para almacenar las notas de un alumno de las asignaturas de un trimestre:

```c
#include <stdio.h>
typedef struct{
    int matematicas;
    int quimica;
    int informatica;
}notas;
// vamos a usar paso por referencia es decir punteros
void darnotas(notas *not);

int main(){

    notas not;
    darnotas(&not);

    printf("La nota de matematicas es: %i.\n",not.matematicas);
    printf("La nota de quimica es: %i.\n",not.quimica);
    printf("La nota de informatica es: %i.\n",not.informatica);

    system("pause");
    return 0;
}

void darnotas(notas *not){
    printf("Introduzca la nota de matematicas: ");
    scanf("%i",&not->matematicas);  //el punto que suele tener se cambia por ->

    printf("Introduzca la nota de quimica: ");
    scanf("%i",&not->quimica);

    printf("Introduzca la nota de informatica: ");
    scanf("%i",&not->informatica);
}
```

```
C:\Users\pc2\Documents\notasClase.exe
Introduzca la nota de matematicas: 7
Introduzca la nota de quimica: 9
Introduzca la nota de informatica: 10
La nota de matematicas es: 7.
La nota de quimica es: 9.
La nota de informatica es: 10.
Presione una tecla para continuar . . .
```

PROGRAMACION EN C: BLOQUE 7

Hasta ahora hemos utilizado la memoria estática y pseudoestática.

MEMORIA ESTATICA

```c
#include <stdio.h>
int main(){
    // memoria estatica
    int i;
    int N = 15;
    int vector[N];

    for(i=0;i<N;i++){
        vector[i] = i;
    }
    for(i=0;i<N;i++){
    printf("%i, ",vector[i]);
    }
    printf("\n");
    system("pause");
    return 0;
}
```

```
C:\Users\pc2\Documents\estatica1.exe
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
Presione una tecla para continuar . . .
```

La memoria reserva espacio para los huecos que le digamos , pero si le mandamos que nos imprima 5 números en este ejemplo se desaprovecha un tamaño de 45 y además si queremos algo más grande , es decir le pedimos 60 números dará un error por exceder el tamaño.

MEMORIA PSEUDOESTATICA

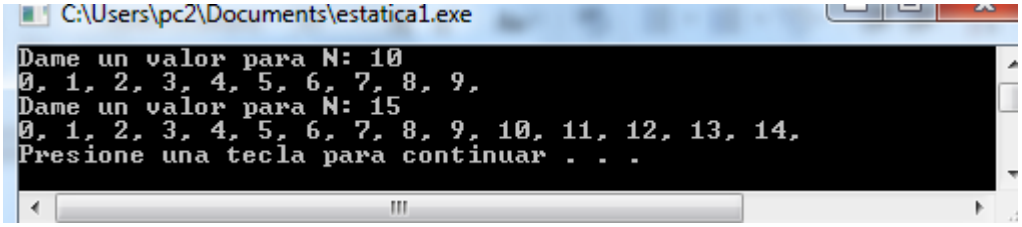
```c
#include <stdio.h>
int main(){
    // memoria PSEUDOESTATICA
    int i;
    int N;

    printf("Dame un valor para N: ");
    scanf("%i",&N);
    int vector[N];

    for(i=0;i<N;i++){
        vector[i] = i;
    }
    for(i=0;i<N;i++){
    printf("%i, ",vector[i]);
    }
    printf("\n");
    system("pause");
    return 0;
}
```

```
C:\Users\pc2\Documents\estatic...
Dame un valor para N: 5
0, 1, 2, 3, 4,
Presione una tecla para continuar .
```

Nosotros decimos el número exacto de memoria que debe reservar mediante el teclado ,podemos elegir como de grande es nuestro vector ,
Pero más adelante no se puede modificar el tamaño.

MEMORIA DINAMICA :

```c
#include <stdio.h>
#include <stdlib.h>
int main(){
    // MEMORIA DINÁMICA
    int i;
    int N;
    int *vector;

    printf("Dame un valor para N: ");
    scanf("%i",&N);

    vector = (int*)malloc(N*sizeof(int));
    // queda crado nuestro vector dinámico
    if( vector == NULL){
        printf("No se a podido reservar la memoria.\n");
    }
    else{
        for(i=0;i<N;i++){
            *(vector + i) = i;
        }
        for(i=0;i<N;i++){
            printf("%i, ",*(vector + i));
        }
        printf("\n");

        printf("Dame un valor para N: ");
        scanf("%i",&N);

        vector = (int*)malloc(N*sizeof(int));
        if( vector == NULL){
            printf("No se a podido reservar la memoria.\n");
        }
        printf("\n");

        printf("Dame un valor para N: ");
        scanf("%i",&N);

        vector = (int*)malloc(N*sizeof(int));
        if( vector == NULL){
            printf("No se a podido reservar la memoria.\n");
        }
        else{
            for(i=0;i<N;i++){
                *(vector + i) = i;
            }
            for(i=0;i<N;i++){
                printf("%i, ",*(vector + i));
            }
            printf("\n");
        }

    system("pause");
    return 0;
    }
}
```



CON MEMORIA DINÁMICA SIEMPRE SE USAN PUNTEROS

NECESITAMOS LA LIBRERÍA STDLIB

Hemos creado un vector dinámico al que nosotros daremos N posiciones , y lo bueno es que si hemos elegido un vector de 5 posiciones podemos hacerlo más grande

Este tipo de memoria suple los problemas de las otras memorias .

La desventaja es que se hace un código más largo , cada vez que hacemos una llamada al malloc realentizamos el procesador

CONSEJO:

cuando sean vectores pequeños usad la estática y para grandes vectores , matrices etc mejor la dinámica.

MALLOC Y REALOC:

MALLOC : asigna un tamaño al vector (**reserva el espacio de memoria suficiente como el que estamos pidiendo**)

El sizeof es una función que lo que hace es dar el tamaño del tipo que tenemos entre paréntesis es decir un entero son 2 bits entonces sizeof(int) son 2 bits en el ejemplo son 2 bits por N , donde N vale 10 ; entonces reserva 20 bits.

1º damos un tamaño
int N = 10;
2º creamos un vector dinamico
int *vector;
3º asignar tamaño a vector dinamico
vector = (int*)malloc(N*sizeof(int));
	Si nuestro vector tiene dos punteros (vector) al asignarlo se pone unos menos en la parte izquierda del igual ,bueno Lo entenderéis con los colores azules.**
	
	OJO : EL (N*sizeof(….)) ese * es un por de multiplicación.
	
	2 punteros :
	1º damos un tamaño
	int N = 10;
	2º creamos un vector dinamico
	int **vector;
	3º asignar tamaño a vector dinamico
	***vector = (int**)malloc(N*sizeof(int*));**
	
	3 punteros :
	int ***vector;
	3º asignar tamaño a vector dinamico
	****vector = (int***)malloc(N*sizeof(int**));**

REALLOC : redimensiona el vector conservando sus valores

La diferencia entre malloc y realloc es que si cogemos malloc vuelve a asignar otro espacio que quizá conserve los valores o quizás no del vector ,no se garantiza , entonces si a nosotros lo que nos interesa es que conserve los valores de dentro utilizamos el realloc que coge el vector ,lo redimensiona y busca un sitio en la memoria donde pueda entrar

Ponemos nombre del vector :

vector = (int*)realloc(vector,N*sizeof(int));

VEAMOS UN EJEMPLO (un poco complejo pero tenemos ya nivel para entenderlo):

Realizar un vector dinámico de 10 elementos aleatorios comprendidos entre 0 y 2 y eliminar todos los números repetidos reajustando el vector.

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
void aleatorio(int *vector, int N);

int main(){
    int i , j,k;
    int N =10;
    int *vector;
    vector = (int)malloc(N*sizeof(int));
    if(vector == NULL){
        printf("No se a podido reservar espacio de memoria.\n");
    }
    else{
        // Para pasar un vector por referencia lo unico que hacemos es poner el nombre ,
        // ni punteros ni corchetes solo el nombre, con el nombre lo que se pasa es solo
        // la direccion de memoria.
        aleatorio(vector,N);
        // algoritmo que elimina numeros repetidos
        for(i=0;i<N;i++){
            j = i +1; // j es un elemento más es decir el 2º e i es el 1º
            while(j<N){
                if(*(vector+i) == *(vector+j)){ // si el vector de la posicion i es = a
                                                // la posicion del vector de la posicion j
                    for(k=j;k+1<N;k++){
                        *(vector+k) = *(vector+k+1); // sustituye el elemento en el que estamos por el sig
                    }
                    N--; //se reduce el vector N a uno menos porque entonces el ultimo elemento ya no lo queremos
                }
                else{
                    j++; // en caso de que no se cumpla avanza j es decir compara el siguiente
                }
            }
```

```
34
35          }
36          // hemos eliminado los elementos repetidos ahora queremos el vector más pequeño
37          // ¿como eliminamos los ultimos elementos de la cola?
38          vector = (int*)realloc(vector,N*sizeof(int));
39          if(vector == NULL){
40              printf("No se a podido reservar espacio de memoria.\n");
41          }
42          else{
43              for (i =0;i<N;i++){
44                  printf("%i, ",*(vector+i));
45              }
46              printf("\n");
47          }
48      }
49
50      system("pause");
51      return 0;
52
53  }
54
55  void aleatorio(int *vector, int N){
56      int i;
57      srand(time(NULL));
58
59      for(i=0;i<N;i++){
60          *(vector+i) = rand() % 3;
61          printf("%i, ",*(vector+i));
62      }
63      printf("\n");
64  }
```

```
C:\Users\pc2\Documents\estatica2.exe
0, 0, 1, 0, 1, 1, 2, 0, 0, 0,
0, 1, 2,
Presione una tecla para continuar . . .
```

CALLOC Y FREE:

CALLOC: se usa para lo mismo que malloc pero también asigna espacio de memoria , tiene un par de diferencias:

no se multiplican los elementos se cambia el " * " por " , "

La diferencia es que malloc asignamos espacio de memoria pero no lo inicializamos mientras que con calloc si que se inicia el espacio de memoria, entonces utilizamos malloc cuando no nos importe lo que halla dentro del vector dinamico porque nosotros lo vamos a reinscribir si o si y calloc se utiliza cuando queramos que nuestro vector se inicie en 0 para el caso del contador , otra cosa entre malloc y calloc , calloc es un pelin mas lento porque además de asignar memoria la inicializa a 0 por eso no siempre utilizamos calloc.

malloc	calloc

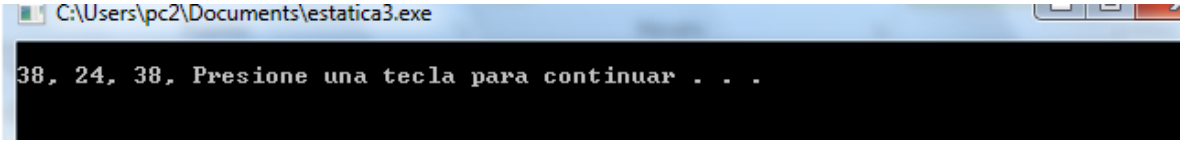
1º damos un tamaño
int N = 10;
2º creamos un vector dinamico
int *vector;
3º asignar tamaño a vector dinamico
vector = (int*)malloc(N*sizeof(int));
	1º damos un tamaño
	int N = 10;
	2º creamos un vector dinamico
	int *vector;
	3º asignar tamaño a vector dinamico
	vector = (int*)malloc(N,sizeof(int));

FREE: libera la memoria dinámica que hemos estado utilizando al final del código (hay que utilizarlo siempre para que no gastemos memoria de más) , pero también se puede utilizar para otra cosa , imaginemos que tenemos un programa muy grande y utilizamos un vector dinámico muy grande y no lo volvemos a utilizar pues entonces cuando sepamos que no vamos a utilizar un vector más veces ponemos el free y así ya podemos dejar que la memoria que ha ocupado se deje libre para otra cosa.

Vamos a ver un ejemplo de todo esto :

Hacer un vector con 100 enteros que estén comprendidos entre 0 y 2 y contar con un vector dinámico cuantas veces aparece cada número:

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
void aleatorio(int vector[], int N);

int main(){

// lo primero a tener en cuanta es saber cuando hace un vector dinamico y cuando estatico
// si en el ejemplo tenemos un valor fijo asignado de 100 elementos entonces será
    int i,op;
    int N = 3; // para 0 , 1 y 2
    int vector[100]; // vector estatico de 100 elementos
    int *contador; // vector contador de manera dinámica
    //1º asiganmos el espacio de memoria al vector dinámico
    contador = (int*)calloc(N,sizeof(int));
    //2º comprobar si se ha conseguido asignar este espacio de memoria
    if(contador == NULL){
        printf("No se a podido asignar la memoria.\n");
    }
    else{
        aleatorio(vector,100); // llamo al vector y le indico el tamaño
        for(i=0;i<100;i++){
            op = vector[i];// registro el valor que tiene el vector en cada momento
            *(contador+op) +=1; // contador tiene en este momento en numero de elementos que tengan las
            // posiciones [0,1,2]
        }
        for(i=0;i<N;i++){
            printf("%i, ",*(contador+i));
        }
    }

    free(contador);

    system("pause");
    return 0;
}

void aleatorio(int vector[], int tam){
    int i;
    srand(time(NULL));

    for(i=0;i<tam;i++){
        vector[i] = rand() % 3;
    }
    printf("\n");
}
```



Si sumamos estos números efectivamente obtendremos 100.

Con esto ya hemos terminado de ver todas las funciones de la memoria dinámica , pero nos quedan por ver otros tipos de elementos que se pueden hacer con la memoria dinámica como estructuras o matrices .

STRINGS DINÁMICOS Y FUNCION EXIT:

STRING DINÁMICO: Básicamente es un string al cual le podemos modificar su tamaño , es exactamente lo mismo que es un vector dinámico a un vector .

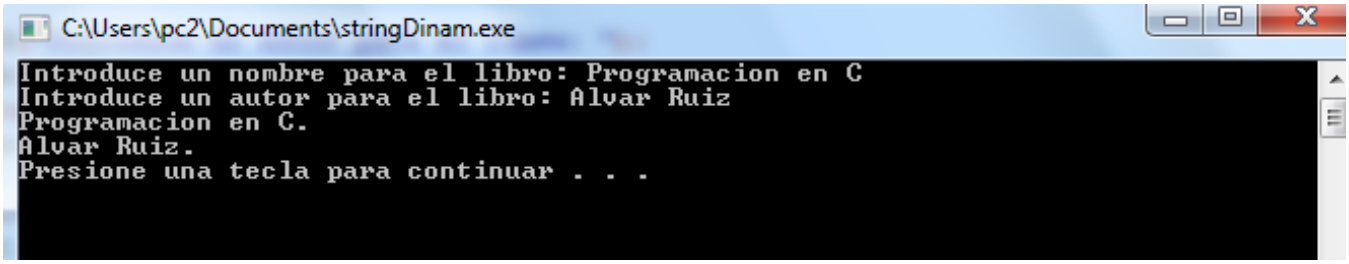
Se deben usar strings dinámicos , en este momento yo he creado un string de tamaño 50 , si escribo 3 , 47 huecos se desperdician , pues si yo uso un string dinámico si uso tres huecos el string ocupa 3 huecos.

FUNCION EXIT : el exit básicamente tiene la función de que si se produce un error sale por ese exit que creemos , es decir en cuanto violemos alguna condición de nuestro programa si disponemos de varios exit nos dira en cual hemos fallado.

EJEMPLO: crea una estructura que represente un libro y usa los strings dinámicos.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct{
    char *nombre; // string dinámico
    char *autor;  // string dinámico
}libro;

void cambio(char aux[100]);

int main(){

    char aux[50]; // string estatico auxilar
    libro PEC;
    printf("Introduce un nombre para el libro: ");
    fgets(aux,50,stdin);
    cambio(aux);

    PEC.nombre = (char*)malloc((strlen(aux)+1)*sizeof(char));// asignamos espacio de memoria al string dinámico
    if(PEC.nombre == NULL){
        printf("No se a podido reservar memoria.\n");
        exit(1);
    }
    // copiamos la cadena
    strcpy(PEC.nombre,aux);//strcpy(destino ,fuente)

    printf("Introduce un autor para el libro: ");
    fflush(stdin);
    fgets(aux,50,stdin);
    cambio(aux);

```

```
34      PEC.autor = (char*)malloc((strlen(aux)+1)*sizeof(char));// asignamos espacio de memoria al string dinámico
35      if(PEC.autor == NULL){
36          printf("No se a podido reservar memoria.\n");
37          exit(1);
38      }
39      // copiamos la cadena
40      strcpy(PEC.autor,aux);//strcpy(destino ,fuente)
41
42      printf("%s.\n",PEC.nombre);
43      printf("%s.\n",PEC.autor);
44
45      // QUE NO SE NOS OLVIDE EL FREE PARA LIBERAR MEMORIA ;P
46
47      free(PEC.nombre);
48      free(PEC.autor);
49
50      system("pause");
51      return 0;
52  }
53
54  void cambio(char aux[50]){
55      int i,temp =0;
56
57      for(i=0;i<100 && temp ==0;i++){
58          if(aux[i]== '\n'){
59              aux[i] = '\0';
60              temp = 1;
61          }
62      }
63  }
```



Los strings dinámicos son una tontería , si nos fijamos lo único que hemos hecho es repetir el código de nombre en autor , así de simple y darnos cuenta de que 1º asignamos un espacio y luego lo copiamos en el vector estático total de tamaño 50.

MATRICES DINÁMICAS :

Una matriz dinámica es una matriz a la cual nosotros podemos redimensionar su tamaño y lo que va a ocupar en memoria .

EJEMPLO : Realizar un programa que copie el contenido de una matriz en otra y todo ello de manera dinámica:

```c
#include <stdio.h>
#include <stdlib.h>

int main(){
    int filas , colum , i ,j;
    int **m1; // necesitamos 2 punteros debido a que una matriz es un vector de vectores
    int **m2; // necesitamos 2 punteros debido a que una matriz es un vector de vectores

    printf("Introduce el numero de filas: ");
    scanf("%i",&filas);
    printf("Introduce el numero de columnas: ");
    scanf("%i",&colum);

    // reservar espacio de memoria :
    // reservar un vector dinámico con filas  y dentro de un elemento del vector creo otro vectror para las columnas
    m1 = (int**)malloc(filas*sizeof(int*));
    if(m1 == NULL){
        printf("No se a podido reservar memoria.\n");
        exit(1);
    }
    for(i=0;i<filas;i++){
        m1[i] = (int*)malloc(colum*sizeof(int));
        if(m1[i] == NULL){
            printf("No se a podido reservar memoria.\n");
            exit(1);
        }
    }

    // asignamos valores para m1

    for(i=0;i<filas;i++){
        for(j=0;j<colum;j++){
            scanf("%i",&m1[i][j]);
        }
    }
    // asignar memoria a m2
    m2 = (int**)malloc(filas*sizeof(int*));
    if(m2 == NULL){
        printf("No se a podido reservar memoria.\n");
        exit(1);
    }
    for(i=0;i<filas;i++){
        m2[i] = (int*)malloc(colum*sizeof(int));
        if(m2[i] == NULL){
            printf("No se a podido reservar memoria.\n");
            exit(1);
        }
    }
    // copiar valores de m1 en m2
    for(i=0;i<filas;i++){
        for(j=0;j<colum;j++){
            m2[i][j] = m1[i][j];
            printf(" %i ",m2[i][j]);
        }
        printf("\n");
    }
    // liberamos memoria
    free(m1);
    free(m2);

    system("pause");
    return 0;
}
```

```
Introduce el numero de filas: 3
Introduce el numero de columnas: 3
1
2
3
4
5
6
7
8
9
 1  2  3
 4  5  6
 7  8  9
Presione una tecla para continuar . . .
```

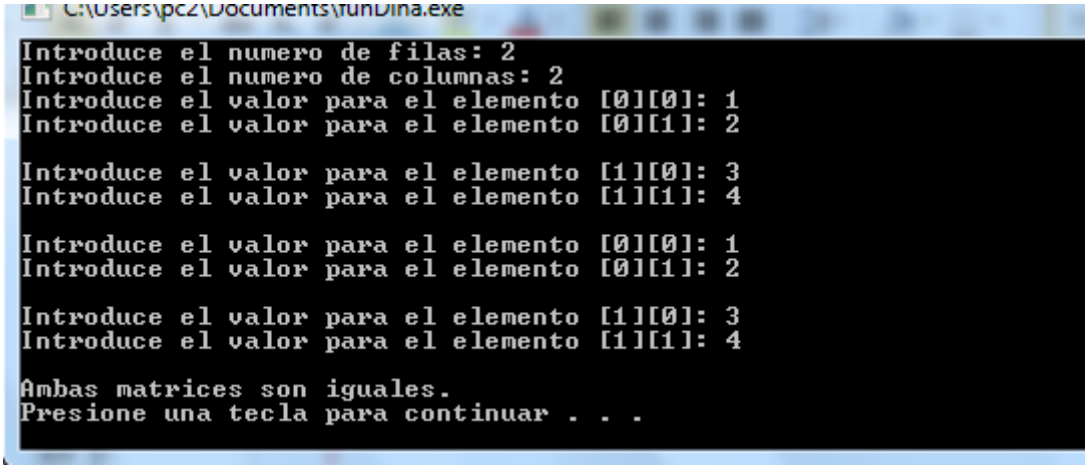
FUNCIONES Y MEMORIA DINÁMICA:

Vamos a usar funciones como las que hemos visto hasta ahora para jugar con la memoria dinámica .

EJEMPLO: COMPROBAR QUE 2 MATRICES DINAMICAS SON IGUALES CON UNA FUNCION.

```c
#include <stdio.h>
#include <stdlib.h>

int reservar(int filas , int colum);
void introduce(int filas , int colum , int **mat);
void comparar(int filas, int colum ,int **m1 ,int **m2);

int main(){

    int filas , colum , i ,j;
    int **m1;
    int **m2;

    printf("Introduce el numero de filas: ");
    scanf("%i",&filas);
    printf("Introduce el numero de columnas: ");
    scanf("%i",&colum);

    m1 = reservar(filas,colum);
    m2 = reservar(filas,colum);

    introduce(filas,colum,m1);
    introduce(filas,colum,m2);

    comparar(filas,colum,m1,m2);

    system("pause");
    return 0;
}

int reservar(int filas , int colum){  // reserva espacio para las matrices
    int i;
    int **mat; // a esta matriz la vamos a asignar memoria
    mat = (int**)malloc(filas*sizeof(int*));
    if(mat == NULL){
        printf("No se a podido reservar memoria.\n");
        exit(1);
    }
    // reservar espacio para el resto de vectores
    for(i=0;i<filas;i++){
        mat[i] = (int*)malloc(colum*sizeof(int));
        if(mat[i] == NULL){
            printf("No se a podido reservar memoria.\n");
            exit(1);
        }
    }
    return mat; // devuelve el tipo creado
}

void introduce(int filas , int colum , int **mat){ // introduce valores en la matriz
    int i,j;
    for(i=0;i<filas;i++){
        for(j=0;j<colum;j++){
            printf("Introduce el valor para el elemento [%i][%i]: ",i,j);
            scanf("%i",&mat[i][j]);
        }
        printf("\n");
    }
}
```

```c
void comparar(int filas, int colum ,int **m1 ,int **m2){
    int i, j;
    int aux = 0;
        for(i=0;i<filas && aux == 0;i++){
            for(j=0;j<colum && aux == 0;j++){
                if(m1[i][j] != m2[i][j]){
                    aux = 1;
                }
            }
        }
        if(aux == 0){
            printf("Ambas matrices son iguales.\n");
        }
        else{
            printf("Las matrices NO son iguales.\n");
        }
}
```



ESTRUCTURAS DINÁMICAS:

Hacer un ejemplo de batallas por turnos entre un héroe y varios malos , los malos se incrementan en 1 cuando un contador del turno de la batalla sea par.

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
// 1º necesitamos la estructura de un heroe
typedef struct{
    int salud;
    int mana;
    int fuerza;
    int magia;
}heroe;

// 2º necesitamos la estructura de un malo
typedef struct{
    int salud;
    int fuerza;
}malo;

heroe yo;
malo *otros; // sera un vector dinamico para que se pueda ir incrementando

void introduceyo();
void introducemalo(int N);
void asignamem(int *N);

int main(){
    int op,i,objetivo,pegar;
    // creamos el contador de la batalla
    int cont =0;
    int N =0; // malos de la batalla
    srand(time(NULL)); // semilla

    // inicializamos nuestro heroe
    introduceyo();
```

```c
    //asignamos memoria a nuestro vector dinámico
    asignamem(&N); // paso por referencia

    // batalla
    do{
        do{ // elige el heroe que quiere hacer
            printf("(1) Atacar.\n");
            printf("(2) Magia.\n");
            scanf("%i",&op);
        }while(op < 1 || op > 2);

        printf("Elige al malo al que quieres pegar.\n");

        for(i=0;i<N;i++){
            printf("Malo %i tiene una vida de: %i.\n",i,otros[i].salud);
        }
        scanf("%i",&objetivo);

        if(otros[objetivo].salud > 0){
            switch(op){
                case 1:
                    pegar = yo.fuerza;
                    printf("He pegado %i.\n",pegar);
                    otros[objetivo].salud -= pegar;
                    printf("La salud del Malo %i es de %i.\n",objetivo,otros[objetivo].salud);
                    break;
                case 2:
                    if(yo.mana > 0){
                        pegar = yo.magia * (rand()%3); // mi magia se multiplica por un numero aleatorio entre 0 y 2
                        printf("He pegado %i.\n",pegar);
                        otros[objetivo].salud -= pegar;
                        printf("La salud del Malo %i es de %i.\n",objetivo,otros[objetivo].salud);
                    }
                    else{
                        printf("No tienes mana !!");
                    }
                    break;
            }
        }
        else{
            printf("Deja que ya te lo has cargado \n");
        }
        printf("Turno de los malos: \n");
        for (i = 0 ; i<N; i++){
            if(otros[i].salud > 0){
                pegar = otros[i].fuerza * (rand() % 3);
                printf("El malo %i me ha pegado %i.\n",i,pegar);
                yo.salud -= pegar;
                printf("Mi salud es de: %i\n",yo.salud);
            }
        }

        if((cont % 2)== 0){
            otros = (malo*)realloc(otros,(N+1)*sizeof(malo));
            if(otros == NULL){
                printf("No se ha podido reservar memoria.\n");
                exit(1);
            }
            introducemalo(N);
            N++;
        }
        cont++;
    }while(yo.salud > 0); // se repite el bucle mientras el heroe este vivo


    system ("pause");
```

```
100     return 0;
101 }
102
103 void introduceyo(){
104     yo.salud = 1000;
105     yo.mana = 100;
106     yo.fuerza = 50;
107     yo.magia =100;
108 }
109
110 void introducemalo(int N){
111     otros[N].salud = 100;
112     otros[N].fuerza = 50;
113 }
114
115 void asignamem(int *N){
116     otros = (malo*)malloc((*N+1)*sizeof(malo));
117     if(otros == NULL){
118         printf("No se ha podido reservar memoria.\n");
119         exit(1);
120     }
121     introducemalo(*N);
122     (*N)++;
123 }
```

Parte 1

```
C:\Users\pc2\Documents\batallas.exe
(1) Atacar.
(2) Magia.
1
Elige al malo al que quieres pegar.
Malo 0 tiene una vida de: 100.
0
He pegado 50.
La salud del Malo 0 es de 50.
Turno de los malos:
El malo 0 me ha pegado 50.
Mi salud es de: 950
(1) Atacar.
(2) Magia.
2
Elige al malo al que quieres pegar.
Malo 0 tiene una vida de: 50.
Malo 1 tiene una vida de: 100.
1
He pegado 0.
La salud del Malo 1 es de 100.
Turno de los malos:
El malo 0 me ha pegado 50.
Mi salud es de: 900
El malo 1 me ha pegado 50.
Mi salud es de: 850
(1) Atacar.
(2) Magia.
2
Elige al malo al que quieres pegar.
Malo 0 tiene una vida de: 50.
Malo 1 tiene una vida de: 100.
1
He pegado 200.
La salud del Malo 1 es de -100.
Turno de los malos:
El malo 0 me ha pegado 50.
Mi salud es de: 800
(1) Atacar.
(2) Magia.
1
Elige al malo al que quieres pegar.
Malo 0 tiene una vida de: 50.
Malo 1 tiene una vida de: -100.
Malo 2 tiene una vida de: 100.
0
He pegado 50.
```

Parte 2

```
La salud del Malo 0 es de 0.
Turno de los malos:
El malo 2 me ha pegado 50.
Mi salud es de: 750
(1) Atacar.
(2) Magia.
2
Elige al malo al que quieres pegar.
Malo 0 tiene una vida de: 0.
Malo 1 tiene una vida de: -100.
Malo 2 tiene una vida de: 100.
2
He pegado 200.
La salud del Malo 2 es de -100.
Turno de los malos:
(1) Atacar.
(2) Magia.
1
Elige al malo al que quieres pegar.
Malo 0 tiene una vida de: 0.
Malo 1 tiene una vida de: -100.
Malo 2 tiene una vida de: -100.
Malo 3 tiene una vida de: 100.
3
He pegado 50.
La salud del Malo 3 es de 50.
Turno de los malos:
El malo 3 me ha pegado 50.
Mi salud es de: 700
(1) Atacar.
(2) Magia.
1
Elige al malo al que quieres pegar.
Malo 0 tiene una vida de: 0.
Malo 1 tiene una vida de: -100.
Malo 2 tiene una vida de: -100.
Malo 3 tiene una vida de: 50.
3
He pegado 50.
La salud del Malo 3 es de 0.
Turno de los malos:
(1) Atacar.
(2) Magia.
1
Elige al malo al que quieres pegar.
Malo 0 tiene una vida de: 0.
```

Parte 3

```
Malo 1 tiene una vida de: -100.
Malo 2 tiene una vida de: -100.
Malo 3 tiene una vida de: 0.
Malo 4 tiene una vida de: 100.
4
He pegado 50.
La salud del Malo 4 es de 50.
Turno de los malos:
El malo 4 me ha pegado 0.
Mi salud es de: 700
(1) Atacar.
(2) Magia.
1
Elige al malo al que quieres pegar.
Malo 0 tiene una vida de: 0.
Malo 1 tiene una vida de: -100.
Malo 2 tiene una vida de: -100.
Malo 3 tiene una vida de: 0.
Malo 4 tiene una vida de: 50.
4
He pegado 50.
La salud del Malo 4 es de 0.
Turno de los malos:
(1) Atacar.
(2) Magia.
2
Elige al malo al que quieres pegar.
Malo 0 tiene una vida de: 0.
Malo 1 tiene una vida de: -100.
Malo 2 tiene una vida de: -100.
Malo 3 tiene una vida de: 0.
Malo 4 tiene una vida de: 0.
Malo 5 tiene una vida de: 100.
5
He pegado 0.
La salud del Malo 5 es de 100.
Turno de los malos:
El malo 5 me ha pegado 50.
Mi salud es de: 650
(1) Atacar.
(2) Magia.
2
Elige al malo al que quieres pegar.
Malo 0 tiene una vida de: 0.
Malo 1 tiene una vida de: -100.
Malo 2 tiene una vida de: -100.
Malo 3 tiene una vida de: 0.
Malo 4 tiene una vida de: 0.
```

Parte 4

```
Malo 5 tiene una vida de: 100.
5
He pegado 200.
La salud del Malo 5 es de -100.
Turno de los malos:
(1) Atacar.
(2) Magia.
2
Elige al malo al que quieres pegar.
Malo 0 tiene una vida de: 0.
Malo 1 tiene una vida de: -100.
Malo 2 tiene una vida de: -100.
Malo 3 tiene una vida de: 0.
Malo 4 tiene una vida de: 0.
Malo 5 tiene una vida de: -100.
Malo 6 tiene una vida de: 100.
6
He pegado 200.
La salud del Malo 6 es de -100.
Turno de los malos:
(1) Atacar.
(2) Magia.
2
Elige al malo al que quieres pegar.
Malo 0 tiene una vida de: 0.
Malo 1 tiene una vida de: -100.
Malo 2 tiene una vida de: -100.
Malo 3 tiene una vida de: 0.
Malo 4 tiene una vida de: 0.
Malo 5 tiene una vida de: -100.
Malo 6 tiene una vida de: -100.
```

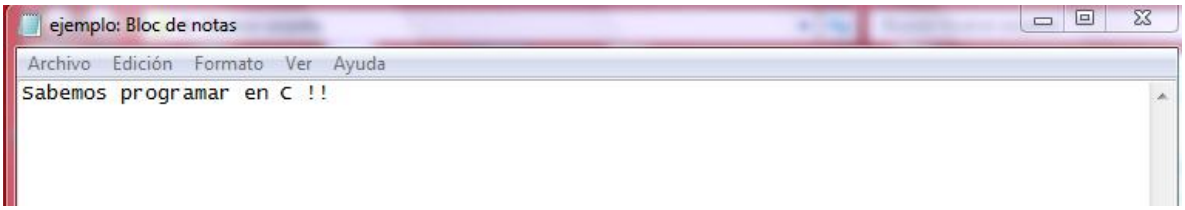
PROGRAMACION EN C: BLOQUE 8

FICHEROS: **Los ficheros son la forma que tenemos de guardar información en el disco duro de nuestro ordenador y leer la información también de nuestro ordenador a nuestro programa.**

Con esta parte de ficheros podemos guardar datos en el disco duro en vez de en la memoria ram como hemos hecho hasta ahora , decir también que ya esto está un poco desfasado ya que para almacenar ahora se utilizan las bases de datos , pero aun asi es interesante saber este tema .

- **Vamos a ver como podemos abrir un fichero y como podemos leer de el :**

1º creamos en el bloc de notas un texto :



2º lo metemos en una carpeta donde también estará nuestro código:

3º hacemos el código:

```
1   #include <stdio.h>
2   int main(){
3
4       char aux;
5       FILE *f;  // declaracion fichero FILE *nombre
6
7       f = fopen("ejemplo.txt","r"); // r modo lectura
8
9       if( f == NULL){
10          printf("No se a podido abrir el fichero.\n");
11          exit(1);
12      }
13
14      // leer desde el fichero
15
16      while(aux != EOF){
17          aux = fgetc(f);  // lee un caracter
18          printf("%c",aux);
19      }
20      printf("\n");
21
22      fclose(f); // equivalente al free y lo
23      //que hace es liberar la memoria
24      // para terminar es muy recomendable
25      //cerrar el fichero cuano no lo usemos más
26
27      system ("pause");
28      return 0;
29  }
```

Declaramos un fichero de la forma que está en la línea 5

FILE *nombre (el * es puntero)

Para abrir en modo lectura línea 7.

De la 9 a la 12 comprobamos que es legible.

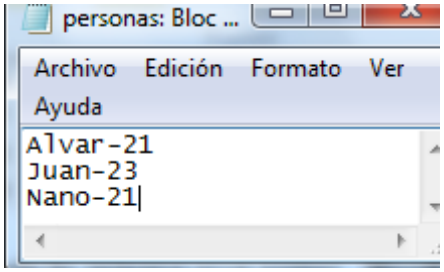
Para leerlo hacemos un bucle el cual lo que hace es meter carácter a carácter en el programa hasta que encuentre EOF que aunque no lo veamos es cuando nuestro texto no tiene más letras.

Para strings:

```c
#include <stdio.h>
int main(){

    char aux[1000]; // cadena de 1000 elementos
    FILE *f;  // declaracion fichero FILE *nombre

    f = fopen("ejemplo.txt","r"); // r modo lectura

    if( f == NULL){
        printf("No se a podido abrir el fichero.\n");
        exit(1);
    }

    // leer desde el fichero

    while(!feof(f)){
        fgets(aux,1000,f);  // se pone el nombre donde antes estaba el stdin
        printf("%s",aux);
    }
    printf("\n");

    fclose(f); // equivalente al free y lo
    //que hace es liberar la memoria
    // para terminar es muy recomendable
    //cerrar el fichero cuano no lo usemos más

    system ("pause");
    return 0;
}
```

Sale lo mismo que antes , !feof(f) --→ mientras no llegue al final imprime el string.

LEER FICHERO Y METER DATOS EN ESTRUCTURA:

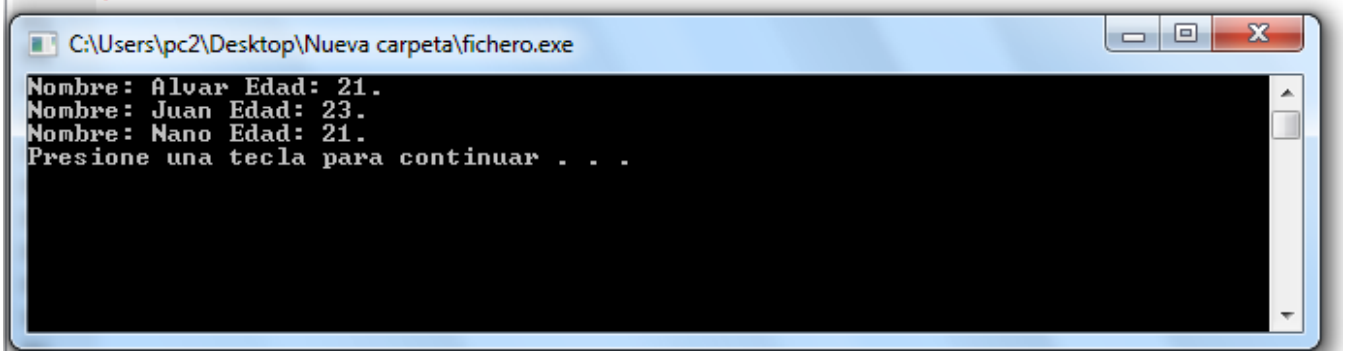


EJEMPLO : COGER LOS DATOS DE LOS ESTUDIANTES DEL FICHERO DE TEXTO Y METERLOS EN UNA ESTRUCTURA:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct{
    char *nombre;
    int edad;
}estudiantes;

estudiantes *estu;

void vaciar(char temp[]);
void copiar(char temp[], int i);


int main(){
    char temp[50];
    int cont =0;
    int i,j;
    char aux;
    FILE *f;
    f = fopen("personas.txt","r");
    if(f == NULL){
        printf("No de a podido abrir el fichero.\n");
        exit(1);
    }

    // reservamos memoria dependiendo del numero de estudiantes
    // tres estudiantes
    while(!feof(f)){
        fgets(temp,50,f);
        cont++;
```

```
33      }
34
35      rewind(f); //pone al inicio el cursor
36
37      estu = (estudiantes*)malloc(cont*sizeof(estudiantes));
38      if(f == NULL){
39          printf("No de a podido abrir el fichero.\n");
40          exit(1);
41      }
42
43      // leemos
44      for(i=0;!feof(f);i++){
45          vaciar(temp);
46          aux = '0';
47          for(j=0;aux != '-';j++){
48              aux = fgetc(f);
49              if (aux != '-'){
50                  temp[j] = aux;
51              }
52          }
53          copiar(temp,i);
54
55          fgets(temp,50,f);
56          estu[i].edad = atoi(temp);// convierte los numeros char en int
57
58          printf("Nombre: %s Edad: %i.\n",estu[i].nombre,estu[i].edad);
59      }
60
61      system ("pause");
62      return 0;
63  }
```

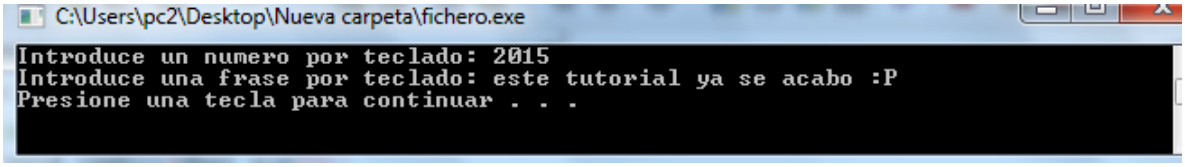
```
64  void vaciar(char temp[]){
65      int i;
66      for(i=0;i<50;i++){
67          temp[i]= '\0';
68      }
69  }
70
71  void copiar(char temp[], int i){
72      int N = strlen(temp) + 1; // cuenta los caracteres dentro de temp
73      // pero no cuenta el caracter de \0
74      estu[i].nombre = (char*)malloc(N*sizeof(char));
75      if(estu[i].nombre == NULL){
76          printf("No de a podido abrir el fichero.\n");
77          exit(1);
78      }
79      strcpy(estu[i].nombre,temp);
80  }
```



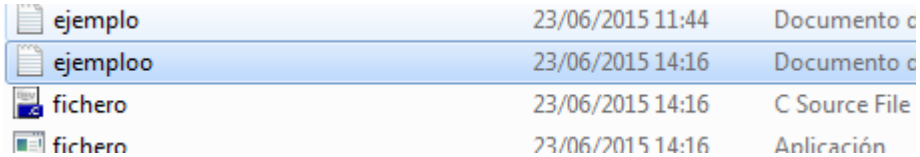
ESCRIBIR EN UN FICHERO DE TEXTO:

Crear un fichero de la nada que contenga un numero y una frase :

```c
#include <stdio.h>
void cambio(char aux[]);

int main(){
    int num;
    char aux[50];
    FILE *f;
    f = fopen("ejemploo.txt","w");
    if(f == NULL){
        printf("No se a podido abrir.\n");
        exit(1);
    }
    printf("Introduce un numero por teclado: ");
    scanf("%i",&num);
    printf("Introduce una frase por teclado: ");
    fflush(stdin);
    fgets(aux,50,stdin);
    cambio(aux);
    fprintf(f,"El numero que has puesto es: %i\n",num);
    fprintf(f,"La frase que has puesto es: %s\n",aux);
    fclose(f);
    system ("pause");
    return 0;
}
void cambio(char aux[]){
    int i;
    for(i=0;i<50;i++){
        if(aux[i] == '\n'){
           aux[i]= '\0';
        }
    }
}
```



Ahora se ha creado un txt de nombre ejemplo con todo:



Y si lo abrimos: